Commodore

Cena 15 tys. zł
nr indeksu 355275

6'93

KEFIR

BOOT X

Miesięcznik Użytkowników Komputerów C-64 i Amiga

MOTOROLA 68060



Video Backup System

Commodore


nr indeksu 355275

Wydawca:
KEBAB - sp. z o.o.
ul. Wojciechowskiego 28
PL - 71-476 Szczecin
telefon (091) 776-74

Redaguje kolegium
w składzie:
Krzysztof Kobus
Patryk Łogiewa
Grzegorz Mikuła
Krzysztof Moroń
Marcin Orłowski
Zbigniew Piotrowicz
Miłosław Smyk
Paweł Sołtysiński

Redaktor naczelny:
Patryk Łogiewa

Szef działu AMIGA:
Krzysztof Kobus
tel. (091) 525-336

Szef działu C-64:
Paweł Sołtysiński
tel. (091) 776-74

Kontakt elektroniczny:
KOBUSKPS@PLSZUS11.BITNET

Projekt okładki:
Tomasz Kuczyński

Commodore


PRENUMERATA

Każdy egzemplarz zakupiony bezpośrednio u nas kosztuje odpowiednio:
numery: 1; 2/3; 4; 5; 6'92 - **9,5 tys. zł**
numery: 7/8'92; ... do 3/4'93 **11 tys. zł.**
(UWAGA! - nakład numeru 6 '92 jest wyczerpany.
numery: 5'93 oraz następne **14 tys. zł.**
Oznacza to, że można zamówić numery zaległe.

Prosimy nie przysyłać do redakcji dowodów wpłat.

Nasze konto:
Pomorski Bank Kredytowy
II Oddział w Szczecinie
numer konta: 368113-25771-136

Podajcie dokładny adres, imię i nazwisko zamawiającego oraz numery egzemplarzy, których wpłata dotyczy na odwrocie każdego z odcinków blankietu wpłaty.

Przy okazji prosimy ponownie o kontakt następujące osoby:
Babula Ewa (?) (nazwisko nieczytelne) z Leszna
Ignaciuk Andrzej z Poznania
Jarmoszewicz Jan z Gryfic,
wpłaciły one po 28.5 tys. na "małą prenumeratę" w marcu ub. roku ale nie podały swojego adresu, natomiast 21.9.92 wpłynęła na nasze konto kwota 96 tys. od **Bacier Adama** (?) (nazwisko niewyraźne). Pan ten napisał, że mieszka przy ul. Targowej 9 nie podając miejscowości oraz celu wpłaty. Ponieważ ogłaszaliśmy już w naszym piśmie te nazwiska bez skutku - prosimy więc naszych Czytelników jeżeli rozpoznają wśród wymienionych swoich znajomych o poinformowanie ich, że chcielibyśmy zrealizować nasze zobowiązania.

I ostatnia prośba - **wszelką korespondencję, wpłaty etc. kierujcie tylko i wyłącznie na adres redakcji.**

REKLAMA

Ogłoszenia drobne od osób indywidualnych (do 10 słów na kuponie wyciętym z III-ciej strony okładki) przyjmujemy bezpłatnie. Ogłoszenia drobne od osób prawnych oraz zawierające powyżej 10 słów są płatne w całości po 1000 zł za słowo.
Ogłoszenia ramkowe (minimalny format 20 cm^2):
1 cm^2 - 4,5 tys. zł, cała strona 2,5 mln. zł,
cała IV strona okładki - 4 mln. zł, 1/2 tej strony - 2,5 mln. zł,
dodatkowy kolor - odpowiednio 50 % drożej.

Ogłoszenia płatne prosimy przesyłać listem poleconym.



Nr 6 **czerwiec 1993**

Witamy serdecznie po raz kolejny.

Tym razem mamy już właściwie wakacyjny numer naszego Kebab'a. Mamy nadzieję, że wśród wakacyjnych przygód znajdzie się trochę czasu i na lekturę komputerową. Spośród rzeczy, które chcielibyśmy Wam przekazać na pierwszym miejscu należałoby wymienić fakt, że cieszy nas bardzo spory odzew na hasło nieustającego konkursu **"O złotą klamkę KEBAB'a"**. Pamiętajcie jednak. Wszystkie zgłoszenia do tego konkursu muszą być udokumentowane. Nie istnieje taka możliwość abyśmy przyznali tak cenną nagrodę jaką jest złota klamka na podstawie nieudokumentowanego zgłoszenia.

Niemniej na podstawie w pełni udokumentowanego zgłoszenia postanowiliśmy przyznać drugą złotą klamkę papierowego KEBAB'a Panu Redaktorowi Naczelnemu Profesjonalnego Magazynu Fanów Komputera Amiga - Markowi Pampuchowi. Ten Pan dokonał na łamach w/w magazynu (AM 1/93 str. 32-24), profesjonalnego testu urządzenia o nazwie KCS Power PC Board. Wśród bardzo wielu ciekawych rzeczy (oraz plusów i minusów jakimi Pan Redaktor był uprzejmy obdzielać firmę KCS), których możemy się z tego "testu" dowiedzieć, znalazły się np. następujące wypowiedzi: "... Licencjonowane wersje MSDOS 5.0... ... zostały potraktowane jak powietrze...", "Profilaktycznie instrukcja radzi... zamiast MSDOS5.0 zalecamy skorzystać z alternatywnego DRDOS-u". No cóż o tym, że KCS Power Board nie chce działać z MSDOS'em 5.0 wiemy od dawna. Widać to zresztą bardzo dobrze w KEBAB'ie nr 5/92 na str. 20. Z pewnością również dlatego, że "instrukcja zaleca..." zainstalowano w wersji 4.5 oprogramowania KCS'a przełącznik (gadget) umożliwiający pełne wykorzystywanie MSDOS'u V5.0... Co do "powietrza", to prawdopodobnie wszystkie dyskietki 5.25 cala zapisane w formacie AT 1.2MB będą tak traktowane, gdyż wszystkie zapisane w formacie 360KB czyli jedynym jaki jest w stanie odczytać KCS z dołączonego "Amigowego" napędu 5.25 działały u nas bezbłędnie... Dalej jednak dowiadujemy się następnych ciekawostek, które rzucają trochę światła na tą tajemniczą sprawę. Otóż okazało się, że Pan Redaktor nie przestudiował instrukcji i nie dokonał "... zalecanej (w takim przypadku) zmiany prędkości obrotowej dysku...". Dla niewtajemniczonych wyjaśnienie jak takiej zmiany dokonać: Należy po pierwsze rozmontować Amigę. Po drugie nie odłączając przewodów zdjąć obudowę (osłonę) z napędu dyskietek i już możemy dokonywać zmian. O ile dokonanie zmiany w kierunku ujemnym jest stosunkowo prostą czynnością gdyż wystarczy przyłożyć palec do powierzchni wirującej dyskietki i wywrzeć odpowiedni nacisk (zmiana będzie wprost proporcjonalna do wartości siły z jaką wywieramy nacisk) o tyle zmiana w kierunku dodatnim wymaga znacznie więcej wprawy i wysiłku. Musimy bowiem tak szybko przebierać paluszkami, że przyłożenie ich do powierzchni dyskietki (w wąskiej szczelinie) lub do koła zamachowego napędu (o ile się doń dostaniemy) spowoduje nie spadek a wzrost prędkości obrotowej. Wśród amatorów nie znaleźliśmy nikogo o wystarczających zdolnościach manualnych... Dużo jeszcze rewelacji można było wyczytać z tego artykułu ale już te wymienione wystarczają w zupełności aby z czystym sumieniem przyznać owo zaszczytne wyróżnienie.

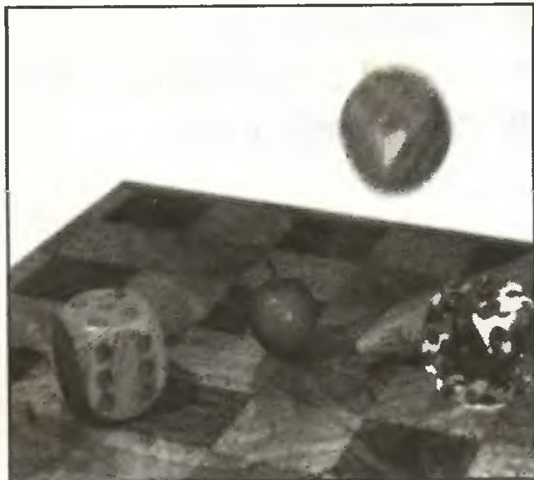
Upominek w postaci dużego (A2), oryginalnego obrazu wykorzystanego jako okładka do KEBAB'a nr 4/92 otrzymuje **Andrzej Zieliński** z Gorzowa Wielkopolskiego. Zarówno laureatowi Złotej Klamki jak i Panu Zielińskiemu składamy gratulacje.

Spis treści :

Po raz kolejny zapowiadane są dodatkowe moduły do Karty Opal Vision. Tym razem producent zapewnia, że dotrzyma obietnicy i z dawna oczekiwane moduły pojawią się w końcu w sprzedaży. Czas pokaże, czy faktycznie tak się stanie. Dotychczas dwa terminy już nie zostały dotrzymane...

Na drugim końcu znajduje się bezpośredni konkurent Opal'a - GVP Impact Vision 24. Ostatnią nowością dotyczącą karty Impact Vision, jest wypuszczenie przez GVP oficjalnego pakietu tzw. "IV24 upgrade pack". Pakiet ten przeznaczony jest dla wszystkich posiadaczy starszych wersji karty. Oprócz nowego ROM'u, dostarczane jest także nowe oprogramowanie na dyskietkach w tym całkowicie zmodyfikowany "MacroPaint" oraz specjalna wersja programu do ray-tracingu i animacji o nazwie "Caligari 24".

Jeśli już mowa o ray-tracingu, to pojawił się również wreszcie oficjalnie nowy Real 3D. Wersja 2.0, której oryginalną wersję mamy nadzieję wkrótce przetestować i opisać, zawiera bowiem praktycznie wszystko co dotychczas wymyślono w dziedzinie tworzenia grafiki i animacji metodami matematycznymi. Efekty uzyskiwane przy jego użyciu są naprawdę doskonałe a fukcje typu "Depth of Focus", "Motion Blur" czy "Lightsampling" pozwalają na uzyskanie obrazów o fotograficznej wierności. Patrz również artykuł dotyczący ray-tracingu wewnątrz nu-



meru oraz obrazki demonstracyjne, które (mamy nadzieję) choć trochę przybliżą to co potrafi nowy Real.

Znany już od dawna Pakiet "Imagine 2" wyposażony jest niestety w bardzo ubogą instrukcję obsługi. W związku z tym, przebojem stała się książka pod tytułem "Understanding Imagine 2" ("Zrozumieć Imagine



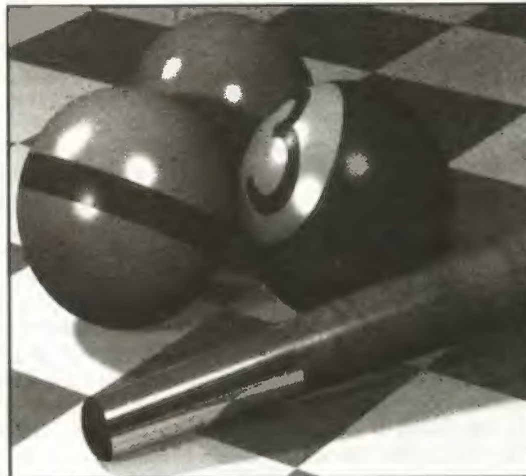
2"). Nakład został bardzo szybko wyczerpany ale ze względu na duży popyt wydawca - Alternate Image, postanowił dodrukować jeszcze trochę egzemplarzy...

Pojawiła się również zapowiadana od pewnego czasu "Scala Infochannel". Ta wersja pakietu przeznaczona jest do obsługi całych sieci telewizyjnych. Za pomocą łączy telefonicznych lub bezpośrednich, satelitarnych można z jednego punktu sterować, uaktualniać dane w wielu stacjach lokalnych np. na drugim końcu kuli ziemskiej...

Znana, amerykańska wytwórnia Warner Brothers tworzy aktualnie nową serię Science Fiction pod tytułem "Babylon 5". Nie byłoby w tym może nic dla nas interesującego, gdyby nie fakt, że efekty specjalne dla tej serii tworzy niewielka firma o nazwie "Foundation Imaging", założona w 1992 roku przez dwóch brytyjczyków. Jeżeli i ten fakt nie jest jeszcze wystarczająco przekonywujący o tym dlaczego informacja ta znalazła się na naszych łamach, to jeśli dodamy, że "Foundation Imaging" tworzy efekty specjalne używając do tego zabawkowych komputerów typu Amiga to sprawa powinna się wyjaśnić. 12 sztuk tego amatorskiego sprzętu wyposażonego w 12 Video Toaster'ów i osiem procesorów 68040, połączonych w sieć z serwerem o pojemności twardego dysku, równej pięć Gigabajtów i łącznej pojemności pamięci równej 324 megabajty jest odpowiedzialne za stworzenie efektów do 20 odcinków serialu.

Przyjemna wiadomość dla fanów GVP. Produkty tej firmy mają... potanieć. Zapewne jednak nie w Polsce...

W dniach 29-31 Maja odbyło się w Goeteborgu największe Demo Party wszechczasów pod nazwą "The Computer Crossroad '93". Party zostało zorganizowane pod patronatem grup "Silents", "Phenomena", "Electra", "Omega", "Cascada", "The CodeBlasters", "Light", "Horizon". Organizatorzy dołożyli wszelkich starań aby party było naprawdę wydarzeniem na skalę



światową. We współpracy ze skandynawskimi liniami lotniczymi SAS, rozprowadzili ogromną ilość materiałów reklamowych informujących jak i skąd można się dostać na imprezę. Przewidziano konkursy na najlepsze demo, grafikę i muzykę dla następujących komputerów: Amiga, C-64, Atari ST i PC-kompatybilne. Jedynie w przypadku PC, podarowano sobie konkurencję muzyczną... Najwyższe nagrody czekały oczywiście na Amigowych koderów, grafików i muzyków. Odpowiednio 20.000 koron za najlepsze demo i po 2.500 koron za najlepszą muzykę i grafikę na Amigę. Aczkolwiek organizatorzy uprzedzali, że wysokość nagród jest uzależniona tylko od sponsorów.

MOC DLA LUDU

Obojętnie w jak szybki procesor wyposażymy nasz komputer, to i tak będzie on zbyt wolny jak na nasze potrzeby. Dla wszystkich cierpiących na głód MIPSów Motorola przygotowała nowy układ oznaczony symbolem MC68060. Jest to superskalarny procesor 32 bitowy czwartej generacji, a właściwie dwa procesory i koprocesor w jednej obudowie. Został on wykonany w technologii HCMOS 0.5 µm i składa się z 2 mln tranzystorów. Zasilany jest napięciem 3.3 V, zredukowany został pobór mocy, który w procesorach o tej wydajności jest już problemem. MC68060 może być taktowany na razie dwiema częstotliwościami - 50 i 66 MHz. W wersji z wolniejszym zegarem osiąga około 77 MIPS, a popędzany częstotliwością 66 MHz daje 100 MIPS i 15 MFLOPS. Jest całkowicie zgodny ze swoim poprzednikiem MC68040.

Architektura superskalarna pozwala na wykonywanie więcej niż jednej instrukcji w takcie zegara. Dlatego też prędkość pracy tego procesora jest wyższa niż częstotliwość zegara taktującego. Posiada on dwa niezależne tory przetwarzania (ang. pipeline) dekodujące i wykonujące instrukcje równolegle. Praca superskalarna może być oczywiście programowo wyłączona dla ułatwienia testowania i poprawiania programów, co jednocześnie zmniejsza pobór mocy. Procesor MC68060 został zoptymalizowany dla większości instrukcji, które wykonywane są teraz w jednym cyklu zegara.

Motorola 68060 osiąga swoją wysoką wydajność między innymi dzięki rozbudowanemu blokowi pobierania instrukcji (ang. instruction fetch controller). Proces pobrania instrukcji do wykonania składa się z czterech etapów, co umożliwia pobranie kolejnej instrukcji z pamięci, podczas gdy wykonywana jest jeszcze instrukcja poprzednia. Dzięki temu procesor nie musi bezczynnie oczekiwać na pobranie rozkazu, co znacznie przyspiesza jego wykonanie. Kolejne fazy pobierania instrukcji to:

- obliczenie adresu instrukcji,

- pobranie rozkazu z pamięci (lub pamięci podręcznej jeśli się on tam znajduje),

- wstępne zdekodowanie,

- buforowanie instrukcji wraz z informacją sterującą do chwili kiedy może zostać ona wykonana.

Rozkazy skoków wykrywane są przed ich wykonaniem, dlatego też instrukcje są pobierane od nowego adresu wcześniej, co ma także niebagatelny wpływ na szybkość pracy procesora. Dane zapisywane do pamięci (także podręcznej) są tymczasowo buforowane, co zmniejsza straty spowodowane powolną pamięcią, gdyż nie jest blokowana praca procesora w oczekiwaniu na możliwość zapisu.

MC68060 posiada zintegrowaną jednostkę zmiennoprzecinkową (podobnie jak MC68040), zgodną z koprocesorami MC68881/882. Obsługuje ona najczęściej używane instrukcje, natomiast pozostałe (m.in. funkcje trygonometryczne) realizowane są programowo. FPU działa równolegle z obiema jednostkami całkowitoliczbowymi, umożliwiając wykonywanie w tym samym czasie dwóch instrukcji całkowitoliczbowych i jednej zmiennoprzecinkowej. Jednostka zmiennoprzecinkowa może zostać programowo wyłączona zmniejszając w ten sposób pobór energii.

WIEŚCI

Procesor MC68060 jest wyposażony w dwie pamięci podręczne (ang. cache), osobne dla instrukcji i danych, z możliwością jednoczesnego dostępu. Każda z nich ma wielkość 8 Kb i organizację czteroblokową z 16 bajtowymi wierszami. Pamięć podręczna może pracować w trybach write-through lub copy-back, identycznie jak ma to miejsce w MC68040. Różnią się one tym, że w trybie write-through procesor dokonuje jednocześnie zapisu do cache'a i RAM'u, natomiast w przypadku copy-back zapis jest dokonywany dopiero w chwili opróżniania (ang. flush) pamięci podręcznej. Tryb copy-back jest oczywiście znacznie szybszy, niestety kosztem nie działania niektórych, źle napisanych programów. Procesor ma możliwość śledzenia aktywności zewnętrznej szyny, co zapewnia spójność cache'a w systemach wieloprocesorowych.

MC68060 posiada wbudowane dwa układy zarządzania pamięcią PMMU (ang. Paged Memory Management Unit), które niezależnie obsługują instrukcje i dane. Strony mogą mieć wielkość 4 lub 8 KB, a każda strona może być indywidualnie zabezpieczona przed zapisem i włączoną lub wyłączoną możliwość cache'owania. Dzięki MMU łatwa jest do zorganizowania ochrona pamięci lub pamięć wirtualna, jednak możliwości te nie są wykorzystywane przez system operacyjny Amigi. Tablice translacji adresów z logicznych na fizyczne znajdują się w pamięci RAM, jednak każde MMU posiada własną pamięć podręczną ATC (ang. address translation cache), w których przechowywane są ostatnio używane opisy stron, co

znacznie przyspiesza pracę tych układów. Układy MMU mogą być wyłączane do celów debugging'u.

Teraz pozostaje tylko czekać na Amigi wyposażone w MC68060. Zgodnie z wypowiedziami szefów Commodore, wkrótce po wejściu nowego procesora do seryjnej produkcji, mają się pojawić moduły procesorowe do Amigi 4000 (T). Oprócz tego powinny pojawić się "dopalacze" niezależnych producentów. Nie należy jednak liczyć na to, że będą one tanie, ale za parę lat...

Konrad Dubiel

SOFTWARE

HexDecBin

W tym miesiącu niespodzianka dla osób piszących programy w assemblerze z wykorzystaniem procedur systemu operacyjnego. Zamieszczony w dziale "Listingi" program źródłowy HexDecBin służy do przeliczania liczb na dowolny z trzech najczęściej spotykanych systemów. Program posiada wiele ograniczeń, jednak ich wyeliminowanie wiąże się z wydłużeniem tekstu źródłowego, na co, biorąc pod uwagę objętość Kebaba, nie mogliśmy sobie pozwolić. Poza tym głównym celem tego artykułu jest zapoznanie Czytelnika z podstawami programowania "pod systemem", a nie kolejna próba rozwiązania problemu przeliczania systemów liczbowych.

Na początek może parę słów o samej obsłudze programu. Po przepisaniu listingu, zasemblowaniu go na dowolnym asemblerze i wreszcie uruchomieniu, na pierwszym z dostępnych w danej chwili ekranów zostanie otworzone okienko. Owo okienko posiada trzy tzw. "string gadgety" podpisane Hex, Dec, Bin. Aby przeliczyć liczbę na pozostałe systemy wystarczy wpisać ją do odpowiedniego gadget'a i nacisnąć klawisz <RETURN>. Wpisując liczbę należy jednak pamiętać o pewnych ograniczeniach:

* Liczba szesnastkowa musi składać się zawsze z ośmiu cyfr. Zatem jeśli chcemy wpisać liczbę 68A, musimy podać 0000068A.

* Cyfry szesnastkowe od A do F muszą być podawane wielkimi literami.

* Liczby dziesiętne traktowane są jako liczby ze znakiem. Zatem wartość szesnastkowa FFFFFFFF, zostanie po przeliczeniu wyświetlona jako -1.

* W związku z powyższym największą liczbą dziesiętną jaką możemy wpisać jest 2147483647.

* Liczby binarne, podobnie jak szesnastkowe, muszą mieć zawsze długość 32 cyfr.

Jeśli którekolwiek z powyższych ograniczeń nie zostanie spełnione program wygeneruje komunikat o błędzie błyskając ekranem. To w zasadzie tyle jeśli chodzi o obsługę programu. Dalsza część tego artykułu będzie techniczną, zatem Czytelnicy bardziej zainteresowani programem, niż tym jak on działa mogą w tym miejscu przerwać lekturę i przystąpić do przepisywania, pozostałych zapraszam do dogłębnej lektury.

System operacyjny Amigi podzielony jest na szereg mniejszych jednostek, tzw. bibliotek (ang. library), oraz urządzeń (ang. devices). Każda biblioteka jest zestawem funkcji (ew. procedur) w pewien sposób ze sobą spokrewnionych, służących określonym celom. Na przykład biblioteka graficzna (ang. graphics) zawiera procedury związane z szeroko rozumianą grafiką, począwszy od czyszczenia ekranu, stawiania punktów, poprzez rysowanie odcinków, elips, wieloboków, do procedur kreślenia tekstu czcionkami o różnych rozmiarach. Natomiast urządzenia systemu operacyjnego zawierają procedury do komunikacji komputera z otoczeniem poprzez tzw. wejście/wyjście, czyli port szeregowy, równoległy, konsolę czy złącze SCSI.

Jako, że nie będziemy dziś pisać oprogramowania dla modemu czy drukarki, urządzenia zostawimy sobie na inną okazję, a zajmiemy się bibliotekami exec, oraz intuition. Funkcje zamieszczone w pierwszej z nich odpowiedzialne są za globalne zarządzanie systemem, a w szczególności obsługą wielozadaniowości (ang. multitasking) i przerwań, dynamicznym przydziałem pamięci, ogólną kontrolą zasobów programowo-sprzętowych naszej Amigi. Biblioteka intuition natomiast zajmuje się całym graficznym interfejsem użytkownika (GUI - graphic user interface), czyli ekranami, okienkami, rozwijanymi menu (przez niektórych określanych mianem repertuarów), gadgetami, słowem tym wszystkim co widzimy na monitorze i za pomocą czego przekazujemy programom komendy, dane, polecenia...

Jak z tego wynika, do obsługi naszego programu zaprzątniemy przede wszystkim intuition, natomiast exec pełnić będzie jedynie rolę pomocniczą.

Zanim jednak skorzystamy z jakiejkolwiek procedury w intuition, musimy tę bibliotekę otworzyć. W tym celu posłużymy się procedurą `OldOpenLibrary` z biblioteki exec. A jak otworzymy exec?

Hex-Dec-Bin
Hex: 0000D020
Dec: 53280
Bin: 00000000000000001101000000100000

Na szczęście nie musimy. Jak już powiedzieliśmy ta biblioteka jest nadzorcą całego systemu, w związku z czym jest zawsze otwarta, a jej adres bazowy znajduje się pod adresem $00000004. Co to oznacza? O-tóż aby umieścić bazę biblioteki w rejestrze a6 procesora musimy wykonać rozkaz:

```
move.l  4.w,a6
```

Tutaj należy powiedzieć, że adres 4 jest JEDYNĄ komórką pamięci komputera zawierającą ZAWSZE daną mającą tę samą interpretację. Oznacza to, że może się zdażyć iż na różnych modelach Amig, bądź nawet tym samym modelu, lecz innej konfiguracji pamięci, wartość (długie słowo) znajdująca się pod adresem 4 będzie różna, lecz ZAWSZE będzie wskazywać nam bazę biblioteki exec.

Mając do dyspozycji bazę jakiejkolwiek biblioteki umieszczoną w rejestrze a6 możemy skoczyć do dowolnej funkcji tejże biblioteki, oczywiście po uprzednim umieszczeniu w odpowiednich rejestrach procesora niezbędnych parametrów wejściowych dla tej funkcji (tymi parametrami na przykład dla funkcji rysowania punktu mogą być współrzędne tego punktu).

Otwieramy zatem bibliotekę intuition umieszczając w rejestrze a1 adres jej pełnej nazwy, po czym skaczemy do wspomnianej już procedury `OldOpenLibrary`. Na wyjściu w rejestrze d0 otrzymujemy bazę biblioteki lub wartość 0 jeśli nastąpił błąd i biblioteka nie może zostać otworzona. Pisząc programy "pod systemem" musimy pamiętać, aby zawsze sprawdzać poprawność wykonania danej funkcji systemowej. Oznacza to, że nigdy nie powinniśmy zakładać, że biblioteka zostanie otworzona. W kodzie programu musimy przewidzieć sytuację, że nastąpi błąd (jedno z praw Murphy'ego głosi, że jeżeli jest jakakolwiek szansa niepowodzenia, to na pewno ono kiedyś wystąpi), i w takim przypadku podjąć pewne kroki informujące użytkownika o tym fakcie. Mając do dyspozycji bazę intuition możemy korzystać z procedur zawartych w tej bibliotece, zatem przechodzimy do otworzenia okna.

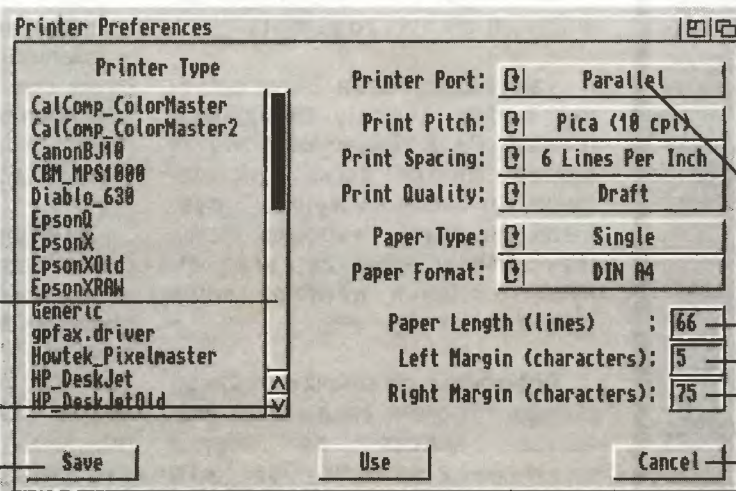
Służy do tego funkcja `OpenWindow`, która wymaga od nas umieszczenia w rejestrze a0 adresu struktury `NewWindow`, opisującej otwierane okno. We wspomnianej strukturze, oprócz takich danych jak wielkość okna, jego współrzędne znajduje się wskaźnik do struktury `Screen`, czyli ekranu na którym chcemy okno otworzyć, o ile nie jest to ekran Workbencha. Pisząc ten program przyjąłem założenie, że nasze okienko będzie otwierało się na ekranie, który jest pierwszy widoczny na monitorze. Adres struktury `Screen` dla takiego ekranu znajduje się pod adresem $3c od bazy biblioteki intuition. Rozwiązanie takie jest bardzo niewskazane, gdyż prawdopodobnie spowoduje zawieszenie systemu w momencie, gdy program, który otworzył swój ekran, zamknie go gdy nasze okienko będzie się na nim znajdować. Poprawnym rozwiązaniem tego problemu byłoby użycie tzw. PublicScreen'ów, ale jest to dopiero możliwe pod systemem 2.0. Przepisujemy zatem adres struktury `Screen` poleceniem:

```
move.l  $3c(a6),$1e(a0)
```

i otwieramy okno skacząc do funkcji `OpenWindow`, sprawdzamy testując d0 poprawność wykonania i w zależności od wyniku generujemy błąd (d0 = 0) albo w komórce Window zapisujemy zwrócony wskaźnik do struktury o tej samej nazwie i przechodzimy dalej.

Teraz w komórce pomocniczej RastPortPtr przechowujemy wskaźnik do struktury `RastPort`, który będzie niezbędny jako argument do procedur graficznych, takich jak na przykład `PrintIText` wypisującej w okienku podany tekst. Wypisujemy zatem tekst, a właściwie teksty Hex, Dec, Bin, oraz procedurą `ActivateGadget` uaktywniamy pierwszy z tzw. StringGadgetów, co przejawia się tym, że w jego środku pojawia się kursor czekający, aż wpiszemy liczbę w stosownym systemie liczenia. Tym sposobem zbliżyliśmy się do głównej pętli programu, zbierającej "bodźce" od użytkownika i stosownie do nich wywołującej odpowiednie podprogramy.

Komunikacja okna z użytkownikiem prowadzona jest przez tak zwany kanał IDCMP. Ów kanał łączy z jednej strony jądro systemu operacyjnego, a z drugiej naszą aplikację poprzez tzw. UserPort. W momencie, gdy użytkownik np. kliknie myszką na jakimś gadgecie, system odbiera tę



Słowniczek:

Gadget - krótko mówiąc, jest to miejsce w obrębie okienka w którym możemy kliknąć myszką aby osiągnąć jakiś cel. Gadgety ogólnie dzielą się na następujące kategorie:

BoolGadget - zwykły gadget mogący przybierać dwie pozycje (włączony - wyłączony) służący najczęściej do zatwierdzania lub nie danej opcji.

StringGadget - po kliknięciu na nim pojawia się kursor umożliwiający podanie aplikacji tekstu (ciągu znaków). Jego odmianą jest IntegerGadget umożliwiający podanie 32-bitowej liczby całkowitej.

ProportionalGadget - poziomy lub pionowy pasek umożliwiający przesuwanie zawartości (najczęściej) listy wyboru.

Omówione tu typy gadget'ów zaliczane są do grupy dostępnych dla programistów. Użytkownik może ponadto korzystać z gadget'ów tzw. systemowych obsługiwanych całkowicie przez system (program nie otrzymuje informacji o ich wybraniu lub zwolnieniu poprzez znaczniki GADGETUP/GADGETDOWN). Należą do nich:

DragGadget - umożliwiający przesuwanie okienek;
DepthGadget - umożliwiający schowanie lub wyciągnięcie na wierzch okna przed stos okienek nakładających się w obrębie ekranu;
SizeGadget - umożliwiający zmianę wielkości okna;
CloseGadget - umożliwiający zamknięcie okna.

Chociaż ostatnie dwa gadget'y nie są obsługiwane w sposób standardowy użytkownik może otrzymać informację o ich wybraniu ustawiając znaczniki NEWSIZE, CLOSEWINDOW.

informację i przesyła do UserPortu "klikniętego" okna wiadomość (ang: message) o zaistniałym fakcie. To wszystko obsługuje system operacyjny.

W tym momencie włącza się nasz program. Odbiera on przesłaną wiadomość, rozpoznaje co było przyczyną jej przesłania, następnie powinien odpowiedzieć, a właściwie potwierdzić systemowi odebraną wiadomość i na końcu wykonać właściwą (swoją) procedurę stosownie do typu odebranej wiadomości. Powiedzieliśm już, że system informuje nas o pewnych zachowaniach użytkownika na przykład o kliknięciu na gadget. Tutaj zapewne pojawia się pytanie o czym jeszcze może system nas poinformować, oraz skąd system wie, które okno jest zainteresowane jakimi sygnałami. Okazuje się, że w strukturze `NewWindow` znajduje się pole `IDCMPFlags`. Właśnie w jego miejsce wpisujemy znaczniki mówiące systemowi jakimi informacjami zainteresowane jest nasze okno. Spójrzmy do naszego listingu. We wspomnianym polu ustawione są następujące znaczniki:

`GADGETUP` - oznacza, że użytkownik chce być poinformowany, gdy gadget w naszym oknie zostanie "zwolniony". W przypadku "StringGadget'ów" termin "zwolnienie" oznacza naciśnięcie klawisza <RETURN> po wpisaniu tekstu do gadgeta.

`CLOSEWINDO` - informuje użytkownika o naciśnięciu tzw. CloseGadget'a znajdującego się w lewym górnym rogu okna.

Jak widać nasze okno odbierać będzie sygnały dwojakiego rodzaju. Poza opisanymi powyżej system oferuje serwis informacyjny o włożeniu/wyjęciu dyskietki ze stacji, wybraniu menu, przyciśnięciu klawisza, oraz innych, o których powiemy sobie przy innej okazji.

Powróćmy do analizy naszego listingu. Program za etykietą MainLoop skacze do funkcji `GetMsg` z adresem UserPort'u okna w rejestrze a0. `GetMsg` sprawdza, czy do wspomnianego portu system przesłał wiadomość i jeśli rzeczywiście port nie jest pusty to w d0 zwraca adres przybyłej struktury `IntuiMessage` (który to przechowujemy w tymczasowej komórce MessagePtr), w przeciwnym wypadku d0 zostaje wyzerowany. Jeżeli zaistnieje ten drugi przypadek program skacze do funkcji `Wait`, czekającej, aż port przestanie być pusty t.j. w konsekwencji działań użytkownika przybędzie "jakaś" wiadomość. Proszę zauważyć, że `Wait` tylko czeka na przybycie wiadomości, nie pobierając jej z portu (tego dokonuje `GetMsg`). Jeżeli doczekamy się wreszcie, aż przybędzie do portu wiadomość to nasz program "znajdzie" się za etykietą Cont. Teraz zadaniem użytkownika jest szybkie rozpoznanie czego wiadomość dotyczy. W tym celu testujemy w strukturze `IntuiMessage` pole nazwane `Class`.

Znajdująca się tam wartość będzie jednym ze znaczników uprzednio ustawionych w polu IDCMPFlags struktury `NewWindow`. Wykonujemy zatem serię porównań w celu rozpoznania przyczyny przysłania informacji i tym sposobem znajdujemy się za etykietą:

GadgetIsUp - oznaczającej zwolnienie któregoś z gadgetów, albo za etykietą:
WindowIsClose - oznaczającej przyciśnięcie CloseGadget'a.

Z racji tego, że ustawiliśmy w programie tylko dwa znaczniki IDCMP (`GADGETUP` i `CLOSEWINDO`) możemy spodziewać się wiadomości spowodowanych jedynie tymi dwoma "bodźcami" od użytkownika - stąd w dwóch liniach programu komentarz "Nie jest konieczne".

Zajmijmy się teraz sytuacją, gdy okazało się, że przybyła wiadomość świadczy o przyciśnięciu gadget'a zamknięcia okna, czyli programem za etykietą WindowIsClose. Najpierw wykonywany jest podprogram ReplyMessage, czyli skok do procedury `ReplyMsg`, potwierdzającej systemowi odbiór wiadomości i jed-

nocześnie informującej go o zakończeniu korzystania z jej zasobów (danych w niej zawartych), a następnie "sprzątamy" po sobie, czyli zamykamy okno (`CloseWindow`), bibliotekę intuition (`CloseLibrary`) i rozkazami:

```
moveq    #0,d0
rts
```

wychodzimy z programu.

A co się dzieje, gdy przybyła wiadomość świadczy o zwolnieniu gadgeta? Jak już powiedzieliśmy procesor skacze wtedy do etykiety GadgetIsUp. Do rejestru a0 ładujemy adres przybyłej struktury `IntuiMessage` i z jej pola o nazwie `IAddress` pobieramy do rejestru a0 adres struktury Gadget zwolnionego gadgeta. W tym momencie wiemy już, który ze StringGadget'ów spowodował przysłanie wiadomości. Teraz z pola `UserData` struktury `Gadget` pobieramy adres procedury jego obsługi i tymczasowo przechowujemy na stosie. Następnie potwierdzamy odebranie wiadomości skacząc do systemowej procedury `ReplyMsg`, zdejmujemy ze stosu adres procedury obsługi naszego gadgeta i skaczemy do niej. Owa procedura dokonuje konwersji liczby pobranej od użytkownika na pozostałe systemy i kończy się skokiem do etykiety MainLoop, czyli sprawdzeniem czy przyszła już kolejna wiadomość od systemu.

Samych procedur konwersji opisywał szczegółowo nie będę, gdyż są one prawie całkowicie niezależne od samego systemu, a celem tego artykułu było przede wszystkim zademonstrowanie Czytelnikom sposobu komunikacji z okienkiem programu. Więcej wiadomości na temat struktur danych, ich poszczególnych pól oraz znaczników można uzyskać przeglądając systemowe include'y.

Krzysztof Kobus

UWAGA! Znacznik IDCMP informujący użytkownika o naciśnięciu CloseGadget'a w rzeczywistości nazywa się CLOSEWINDOW, a nie jak było napisane w artykule CLOSEWINDO.

Zmiana została wprowadzona celowo, aby asemblery nie rozróżniające małych i dużych liter nie generowały komunikatów o dwukrotnej definicji tego symbolu i nazwy procedury zamykającej okno: CloseWindow.

RAM-expansion

SOFTWARE

Pisząc program często chcielibyśmy wykorzystać maksymalnie dostępną na danym komputerze pamięć. Jeśli program ma pracować w środowisku DOS czy WorkBencha to nie ma dużego kłopotu: wystarczy sprawdzić ilość dostępnej wolnej pamięci np. używając AVAILMEM z exec.library (patrz Kebab 5/92) a potem ją zaalokować używając np. ALLOCMEM. Przysłowiowe "schody" zaczynają się gdy nie zamierzamy w programie korzystać z usług systemu operacyjnego lub chcemy go nawet "unieszkodliwić" (bo przykładowo naszym programem będzie całodyskowe demo i system będzie tylko "przeszkadzał") a mimo to chcielibyśmy wiedzieć na ile bajtów RAMu możemy liczyć.

W praktyce okazuje się że całkowite traktowanie systemu operacyjnego po macoszemu skomplikowałoby nam trochę życie i zmuszeni bylibyśmy momentami pisać procedury które "od lat" zaszyte są w ROMie Amigi. Mam na myśli procedurę TypeOfMem z exec.library, którą wykorzystamy do określenia czy dana Amiga ma rozszerzenie pamięci, a jeśli tak, to jakie (przy czym jako rozszerzenie potraktujemy każdą pamięć powyżej standardowej wielkości 512KB).

Otóż procedura TypeOfMem zwraca nam typ pamięci, umieszczonej pod adresem podanym jako jej parametr wejściowy (w rejestrze A1), przy czym typ określany jest analogicznie jak przy procedurach ALLOCMEM czy AVAILMEM. Ponieważ TypeOfMem jako typ zwraca tylko CHIP lub PUBLIC (czyli albo SLOW albo FAST) co dla nas nie jest wystarczającą informacją, skorzystamy więc z faktu, że TypeOfMem zwraca zero gdy podany przez nas adres nie leży w znanym systemowi obszarze pamięci, przy czym, jako taki traktowany jest także ROM!.

Uwaga! Zauważcie, że powyższe rozwiązanie pozwala nam jedynie określić adres dodatkowej pamięci, nie zaś jej rozmiar, lecz ponieważ nie spotyka się rozszerzeń mniejszych niż 512 KB więc bez obaw można szaleć po całym tym obszarze. Myślę, że powyższe informacje wystarczą już wszystkim zainteresowanym do napisania odpowiedniego programu, zaś pozostałych odsyłam do gotowego rozwiązania. Zwróćcie jeszcze uwagę, że nie zawsze jako adres dla funkcji TypeOfMem możemy podać fizyczny adres początku danego rozszerzenia pamięci, ponieważ jest on z reguły zarezerwowany na użytek systemu (w programie adres $80000 jest poprawny ponieważ jest to kontynuacja pamięci CHIP nie zaś odrębny obszar!).

Przykładowo: jeśli chcemy tą funkcją sprawdzić podstawowe 512KB RAMu (od adresu $0000 do $7ffff) to TypeOfMem zwróci nam wartość różną od zera dopiero dla adresów od $420 (ang. lower bound) w górę (sprawdźcie!), bowiem początek tego obszaru zarezerwowany jest między innymi na wektory (np: przerwań czy stanów wyjątkowych) procesora. Zamieszczona procedurka (Listing #1) zwraca

adres pierwszego znalezionego rozszerzenia pamięci (w kolejności CHIP/SLOW/FAST) w rejestrze D0 lub wartość zero (0) gdy takowego rozszerzenia nie stwierdzono. Ponadto w praktyce okazało się, że z punktu widzenia hardware (pomijając najstarsze wersje) pamięć SLOW może być traktowana na równi z CHIP, tzn. możemy umieszczać tam sample czy grafikę pamiętając jednak, że wszystkie te dane będą widziane jakby znajdowały się w rozszerzeniu CHIP RAMie! Np: mając sample pod adresem $C28200 powinniśmy traktować je jakby były pod adresem $A8200.

Metodą alternatywną do powyższej, jest wykorzystanie struktury Memory List zawartej w exec.library. Struktura taka istnieje dla każdego obszaru pamięci RAM "widzianego" przez system operacyjny. Zawiera on między innymi adres początkowy danego obszaru pamięci jak i jego adres końcowy. Ponadto, znajdziemy tam adres wskazujący ciąg znaków (string) opisujących nam dany rodzaj pamięci (Fast/Chip). Te trzy parametry wystarczą nam w zupełności do napisania kolejnego programiku (listing #2), który wypisze nam na standardowym wyjściu (patrz poprzednie numery Kebab'a), pobrane z MemListy interesujące nas dane.

Mr.Soft / W.F.M.H.

ZE SCENY

PADUA

Grupa do której należę, nie jest specjalnie prosta do opisania. Ponieważ miałem już okazję być w innych grupach (Quartet i Science 451) - trochę zdążyłem już zaobserwować zmian zachodzących w ludziach w miarę ich dorastania. O ile się o to nie zadba - zwykle następuje naturalny "zgon" grupy. Tak było w przypadku Quartetu (jedni przeszli na Amigi, drudzy po prostu zaczęli dorośleć, itp.) i Science 451 (tego wzięli do wojska, inni się pożenili). W efekcie grupa albo znikała na zawsze albo odradzała się pod tą samą nazwą ale z innymi, zwykle nowymi na scenie ludźmi.

Kiedy przewidując koniec działalności Science 451 przeniosłem się do grupy Padua, natrafilem tam na wyjątkowo dobre towarzystwo. W grupie znalazło się wiele osób, które przebyły drogę podobną do mojej. Jej trzon stanowią bowiem dawni członkowie grupy Beastie Boys. Dołączyłem do nich zaraz po ukazaniu się ich dyskowego dema TORTURE.

W owym czasie grupa zaczęła powoli nawiązywać kontakty z firmami wydającymi oprogramowanie rozrywkowe (gry) i po konkurencyjnych cenach zaczęto pisanie na zamówienie konwersji gier z Amigi na C-64 i odwrotnie (np. Conquestador). Generalnie argumentem przetargowym była wysoka jakość naszych produkcji oraz szybki czas realizacji. Było to możliwe dzięki podziałowi pracy w zespole. Raz zdarzyło się, że ktoś nawalił (konkretnie LUBBER) i Padua musiała spłacić firmie tzw. fine, czyli przewidzianą w umowie karę za niedotrzymanie terminu zdania gotowego programu.

Sytuacja zaczęła się powoli zmieniać w chwili, gdy kilku członków grupy pokończyło szkoły i trafiło do wojska lub, tradycyjnie, się pożeniło (następni w tym roku...). Gdy tylko to zauważyłem, przy najbliższym spotkaniu z Ano (Padua HQ) poruszyłem tę sprawę i naświetliłem dobrze znane mi następstwa. Spowodowało to dużą zmianę w polityce wewnętrznej grupy (liczącej wówczas około 25 osób z różnych państw). Przede wszystkim zrezygnowano z powszechnej wydajności na korzyść spraw osobistych każdego z członków Padua. Dzięki temu nikt się już nie musiał stresować w stylu "niczego nie zrobiłem od 2 miesięcy - na pewno wyrzucą mnie z grupy". Wręcz przeciwnie: ustalono, że należy położyć większy nacisk na rozwijanie więzów przyjaźni (co się może jeszcze nie raz w życiu przydać). Zadanie raczej niebagatelne, mając na względzie liczbę krajów wchodzących w rachubę. Kilka razy do roku Padua organizuje meetingi dla wszystkich jej członków, gdzie np. padają propozycje np. "dnia bez słowa na temat komputerów".

To wszystko nie oznacza jednak zaprzestania działalności. Nadal kilka osób pracuje nad konwersjami gier, kilka tworzy specjalistyczne oprogramowanie do modemów telefonicznych

i Packet Radio (kilku ostatnio przesadziło i mieli problemy w związku z nielegalnym poruszaniem się po sieciach komputerowych). Nadal prężnie pracują swapperzy. Od 1,5 roku (!) powstaje nowe demo, czyli TORTURE II, ale mając na względzie wymienione powyżej warunki - trochę to "jeszcze" potrwa. A oto aktualny spis członków grupy:

ANONYM (koder C-64 i Amiga)
LEONARDO (koder C-64)
FRANKY (koder C-64 i Amiga; edytor magazynu Update)
CREAT (grafik C-64 i Amiga)
BULLDOG (muzyk C-64)
ROMAN (grafik C-64 i Amiga; edytor magazynu Update)
ARENA (grafik C-64)
HOBBES (swapper C-64 i Amiga)
HORNET (koder C-64 i Amiga)
PIXEL (grafik C-64)
DIABOLO (koder C-64)
MAD (grafik i koder C-64 i Amiga) DEATHLOK (sysop BBS "Blurred Reality" tel. +1-609-455-3492 [USA])
LOOP (swapper C-64 i Amiga)
THE CHIEF (public relations manager)
JADAWIN (koder i muzyk C-64)
ALCHEMIST (sysop BBS "The Dungeon" tel. +1-214-503-7182 [USA])
POLONUS (koder C-64, Packet Radio BBS: 26.840 LSB POL1PL @ ITA130-8)

Chcącym powymieniać się oprogramowaniem zalecam kontakt na te adresy:

Hobbes
Andreas Ryck
Titlisweg 25 A
D-1207 Berlin
GERMANY

Loop Gerorg
Postfach 400328
D-44737 Bochum
GERMANY

Paweł "POLONUS" Sołtysiński

A w Bajtku pisali...

Pewnego razu robiąc porządki w piwnicy znalazłem tajemniczą paczkę. Po rozpakowaniu okazało się, że ów pakunek zawiera kompletne roczniki Bajtka. Jako że w czasach, gdy Bajtek był w zasadzie jedynym (pomijając Komputer, który pojawił się nieco później) źródłem wiedzy informatycznej, dostępnym dla szerokiej rzeszy wszelkiej maści maniaków komputerowych, nie byłem z racji posiadanego ATARI 800XL, zagorzałym czytelnikiem klanu Commodore, pomyślałem, że warto nadrobić zaległości i przeczytać co wtedy o Amidze pisano. Jak pomyślałem tak też zrobiłem a najciekawszymi wyjątkami z przeczytanych artykułów chciałbym niniejszym podzielić się z Czytelnikami Kebaba.

Bajtek 3-4/86, "AMIGA kontra ATARI ST", Borys Schrade na podst. "Happy Computer".

(...) Obie firmy (Commodore i Atari - przyp. red.) zdecydowały się na dyski 3.5 calowe. Wmontowana lub osobno dostępna stacja pamięci dyskowej Amigi ma pojemność 880 kilobajtów. Atari poleca na razie dwa typy stacji: jeden używający dysku jednostronnie (360 kilobajtów) i drugi, obustronny (720 kilobajtów). Oba omawiane komputery różnią się znacznie czasem wczytywania spisu treści jednego dysku; Amiga potrzebuje na wczytanie spisu treści jednego dysku 9 sekund, Atari ST tylko 3 sekundy. (...) Amiga ma bardzo dobrą grafikę: Można w niej mieszać kolory. Obydwa komputery mają rozdzielczość 640x400 punktów (w dwu kolorach). (...)

* * *

Bajtek 1/87, "Centrum obliczeniowe i gryzonie czyli AMIGA", M.S. (Michał Silski).

(...) Mało tego, AMIGA posiada w swym wnętrzu trzy specjalizowane procesory, pieszczotliwie nazwane Agnes, Daphne i Portia, zajmujące się grafiką, animacją, dźwiękiem, portami we/wy i czym tylko jeszcze się da, pozostawiając głównemu procesorowi praktycznie tylko obliczenia. (...) Rozdzielczość ekranu graficznego wynosi od 320x200 pkt. do 640x400 pkt., przy 4096 kolorach, co najciekawsze, możliwych do jednoczesnego otrzymania na ekranie! Poza grafiką o niewiarygodnej jakości AMIGA pozwala nam definiować znane ze starszych modeli "duszki" (sprites), tyle że ich ilość i wielkość nie podlega już takim ograniczeniom, jak kiedyś. Duszków jest teoretycznie 8, ale w rzeczywistości możemy na ekranie zobaczyć dowolną ich liczbę. Jeżeli ktoś nie interesuje się grafiką, AMIGA oferuje doskonały tryb tekstowy (80 znaków * 25 wierszy) oraz ogromne możliwości dźwiękowe. Cztery kanały cyfrowe syntezatora pozwalają tworzyć dowolne dźwięki bez żadnych ograniczeń. Można odtwarzać muzykę przesłaną cyfrowo z Compact-Disku, można generować doskonale brzmiącą ludzką mowę, można imitować instrumenty lub orkiestry, można wreszcie wprowadzić do pamięci komputera cyfrowy zapis dowolnej muzyki, korzystając z wbudowanego wejścia analogowo-cyfrowego (tzw. digitizer). (...) Niezrozumiały jest też długi czas dostępu do katalogu dysku - przy organizacji wewnętrznej komputera z użyciem 25 kanałów DMA czas ten powinien być niezauważalnie krótki. (...)

* * *

Bajtek 3/88, "AMIGA superkomputer pod strzechy", Dominik Falkowski.

(...) Nowatorstwo Amigi 2000 polega na współpracy dwóch systemów: INTEL i MOTOROLA czyli MS-DOS i AMIGA-DOS lub jak kto woli IBM PC/XT/AT i AMIGA. Współpracę tę umożliwią bridgeboard czyli karta koordynująca współpracę obu systemów. Bridgeboard umożliwia zastosowanie tysięcy programów sprawdzonych na komputerach IBM. Amiga 2000 umożliwia również tzw. multitasking czyli obsługę kilku programów jednocześnie. Powoduje to, że bez najmniejszych kłopotów możemy połączyć kilka programów i korzystać z nich jak gdyby to był jeden oryginalny program. Amiga 2000 to również fenomenalna grafika. Procesor MOTOROLA 68000 wspomagany przez 3 pozostałe koprocesory pozwala na tworzenie płynnie animowanej grafiki trójwymiarowej, definiowanie poruszających się obiektów (znane z komputerów domowych sprite'y). Maksymalna rozdzielczość wynosi 640x512 punktów (wykorzystujemy wtedy 16 kolorów z palety 4096 kolorów). Pozostałe tryby pracy procesora graficznego to 320x256 (32 kolory), 320x512 i 640x256. Jest więc z czego wybierać. (...)

Wyczytał i komentarzami nie opatrzył: Krzysztof Kobus

IFF od środka - cz. II

SOFTWARE

W dzisiejszym odcinku naszego krótkiego cyklu poświęconego formatowi IFF przyjrzymy się dokładniej jego odmianie służącej do zapisu animacji - IFF ANIM

Animacja komputerowa to dziedzina, która ostatnimi czasy robi szaloną wręcz furorę. Teledyski, filmy i reklamy telewizyjne pełne są generowanych za pomocą komputerów sekwencji - od konwencjonalnych animacji stworzonych przy użyciu programów pokroju Deluxe Paint'a, poprzez *ray tracing*, na nieco nadużywanym *morphingu* skończywszy.

Amiga, która z założenia była projektowana jako komputer mający dysponować niezwykle szybką grafiką, obecnie przeżywa swoje najlepsze dni. Powstaje masa specjalistycznego oprogramowania, swoją jakością nierzadko przewyższającego software dla stacji roboczych (myślę tu o Real'u 3D V2 i Morph Plus'ie). Tak się szczęśliwie składa, że amigowy format zapisu animacji firma Sparta ustaliła na długo przed rozpoczęciem się owej gorączki, co pozwoliło uniknąć niezwykle cenionej w świecie pecetów mnogości standardów. Zanim jednak zagłębimy się w szczegóły struktur IFF ANIM,

o kompresji animacji w ogólności słów parę...

Konieczność przechowania kilkudziesięciu obrazów dla każdej sekundy animacji zmusza nas bądź do posiadania pojemnych pamięci, bądź do stosowania efektywnych algorytmów kompresji. Ponieważ pamięć wciąż jeszcze kosztuje drożej niż wymyślanie algorytmów (nad czym jako programista szczerze ubolewam), miast zakupów w gatunku kolejnego 2GB HD lub 32MB RAM, dużo powszechniej stosuje się programowe techniki usuwania redundantnych danych.

Gdy przychodzi do wyboru metody jaką tego dokonamy, najprostszym rozwiązaniem wydaje się być kompresja kolejnych klatek za pomocą czegoś, co da się potem w rozsądnym czasie przywrócić do postaci wyjściowej. Na szybkiej Amidze mógłby to być *Imploding*, na wolniejszej *Run Length Encoding*. Praktyka pokazała jednak, że osiągnięte zyski, szczególnie w przypadku bardziej skomplikowanych obrazów nie są zadowalające.

Na szczęście nie trzeba było błyskotliwości geniusza, aby spostrzec iż jest to ogromne pole do popisu dla metod kodowania względnego (patrz Kebab 11-12/92). Następujące po sobie obrazy są przecież na ogół zbliżone wyglądem i nawet w przypadku bardzo drastycznych zmian z klatki na klatkę, znaczny procent powierzchni ekranu pozostaje w bezruchu. Tak więc wystarczy zapamiętywać tylko owe zmiany, w miarę możliwości czymś je dodatkowo pakując. Tak działają wszystkie nowoczesne techniki kompresji animacji, wliczając w to MPEG - niezmiernie bardziej zaawansowany niż IFF ANIM, lecz też w znacznej mierze bazujący na kodowaniu różnic między kolejnymi klatkami (jeśli Czytelnicy wykażą odpowiednie zainteresowanie - listy - omówimy sobie również MPEG).

Powyższa technika ma jeszcze tę zaletę, iż nie musimy dekompresować za każdym razem całości obrazu, lecz tylko jego zmienioną część. W ogromnej większości przypadków poprawia to znacznie prędkość odtwarzania animacji.

Jaka by jednak ta poprawa prędkości nie była - zawsze znajdzie się sekwencja, której nie da się zdekompresować podczas okresu wygaszenia wiązki kreślącej obraz na ekranie monitora. Narażeni bylibyśmy w takim przypadku na "skoki" i "rwanie się" obrazu (patrz format QUICKTIME dla komputerów Macintosh i PC). Aby uniknąć podobnych ekscesów stosuje się powszechnie tzw. *double-buffering*, znany jeszcze z czasów Atari 800XL i C64, gdzie podczas wyświetlania jednej klatki drugą dekompresuje się do specjalnego bufora, by w chwili gdy jest gotowa błyskawicznie ją wyświetlić. Takie podejście do sprawy wymusza jednak kodowanie różnic nie pomiędzy następującymi po sobie obrazami, lecz pomiędzy aktualnym, a tym który pojawił się dwie ramki wcześniej. Trudno jednak nazwać to komplikacją, szczególnie jeśli zauważyć, iż IFF ANIM umożliwia zapis obu wspomnianych typów animacji.

FORM ANIM

rozpoczyna się od struktury FORM ILBM (omówionej dokładnie w poprzednim Kebabie) - czyli zwykłego obrazka, który jest pierwszą klatką naszej animacji. Każda szanująca się "oglądaczka" do grafiki pokaże nam przynajmniej ową klatkę, co umożliwia łatwą identyfikację bez potrzeby ładowania całości pliku.

Następnie zapisane są już chunki będące informacjami o zmianach, które następują z obrazu na obraz. Schematycznie wygląda to tak:

```
FORM ANIM
. FORM ILBM      pierwsza klatka
. . BMHD  zapisana jako "normalny"
. . CMAP                   obrazek
. . BODY
. FORM ILBM          druga klatka
. . ANHD
. . DLTA
. FORM ILBM        trzecia klatka
. . ANHD
. . DLTA
...           itd.
```

Oczywiście jest to schemat bardzo ogólny, bo jak powinniście wiedzieć z poprzedniego numeru naszego pisma, ilość i rozłożenie chunków w pliku IFF są praktycznie dowolne - dbać należy tylko o to, aby zachowana była kolejność klatek.

ANHD (ANimation HeaDer)

to chunk będący odpowiednikem BMHD w obrazku statycznym. Podobnie jak i on, ma za zadanie dostarczać dodatkowych informacji, które mogą okazać się pomocne bądź niezbędne do zdekodowania klatki animacji. A oto jego zawartość:

```
00 UBYTE operation
01 UBYTE mask
02 UWORD w,h
06 WORD  x,y
10 ULONG abstime
14 ULONG reltime
18 UBYTE interleave
19 UBYTE pad0
20 ULONG bits
24 UBYTE pad[16]
```

i znaczenie poszczególnych pól:

operation - metoda kompresji danych. Tu mała dygresja. Poprzez lata doskonalono zarówno prędkość odtwarzania jak i techniki pakowania animacji, co zaowocowało możliwością napotkania w tym bajcie różnych wartości. Wszystkie formaty o numerku mniejszym od pięciu to już właściwie historia, dla porządku wymienię je jednak bez wyjątków:

0 - brak kompresji,
1 - tryb XOR ILBM,
2 - tryb Long Delta,
3 - tryb Short Delta,
4 - ogólny tryb short/long Delta,
5 - tryb Byte Vertical Delta (praktycznie jedyny spotykany)
7 - tryb short/long Vertical Delta,
8 - format używany przez ASDG, 74 - (ASCII "J") zarezerwowany przez Erica Grahama (brak innych danych)

Pomimo tej pozornej mnogości formatów, współczesne programy odtwarzające powinny w zasadzie przejmować się jedynie ANIM5 i ANIM7. Choć obecnie ten pierwszy dominuje i jest jedynym generowanym przez programy komercyjne (z małym wyjątkiem dla pewnej prehistorycznej wersji Videoscape), to w bliskiej przyszłości, w miarę upowszechniania się Amig wyposażonych w chipset AGA, należy liczyć się ze wzrastającą popularnością ANIM7, który na takowym sprzęcie daje istotne różnice w prędkości odtwarzania (na plus, naturalnie). Nasz opis chunku DLTA będzie dotyczył tego występującego w ANIM5, jako najbardziej rozpowszechnionego (wiadomość dla posiadaczy A1200 i A4000 - format ANIM7 mamy już "na tapecie" i wkrótce możecie spodziewać się artykułu na jego temat). Wartość zerowa w polu operation oznacza, iż należy spodziewać się chunku BODY, a nie DLTA. Sytuacja taka wystąpi na przykład wtedy, gdy ANHD pojawił się w obrębie FORM ILBM opisującego pierwszą klatkę. Wówczas informacje w nim zawarte mogą dotyczyć czasu wyświetlania owej klatki.

mask - tylko dla trybu XOR. Poszczególne bity determinują czy w odpowiednim bitplanie nastąpiła zmiana.

w,h - tylko dla trybu XOR. Szerokość i wysokość danych zawartych w BODY.

x,y - tylko dla trybu XOR. Pozycja prostokąta zdefiniowanego powyżej. Pola w, h, x, y służyć miały przyspieszeniu dekompresji w sytuacji gdy tylko pojedynczy fragment ekranu ulega zmianie, idea bierze jednak w łeb w chwili gdy animujemy, dajmy na to, piksel w prawym-górnym rogu ekranu i podobny piksel w lewym-dolnym.

abstime - obecnie nieużywane. Określa ilość czasu, która powinna upłynąć od chwili wyświetlenia pierwszej klatki animacji w jednostkach równych 1/60 sekundy.

reltime - czas, który powinien upłynąć od chwili wyświetlenia poprzedniej klatki w jednostkach równych 1/60 sekundy.

interleave - pole informujące nas o tym, którą klatkę wstecz modyfikują dane w chunku DLTA. Wartości 0 i 2 oznaczają w tym przypadku to samo i są najczęściej (o ile nie jedynie) spotykane - umożliwiają szybki i wygodny *double-buffering.* Jedynka oznaczałaby modyfikację aktualnie wyświetlanych danych, co w niektórych zastosowaniach może być pożądane.

pad0 - bajt niewykorzystany, wyrównuje offset (przesunięcie) w strukturze do długiego słowa.

bits - bity obecne w tym polu zawierają informacje dotyczące różnych trybów pracy metod piątej, szóstej i siódmej. Obecnie używane jest sześć spośród nich i definiuje się je następująco:

bit #	0	1
0	short data	long data
1	set	XOR
2	oddzielne informacje dla każdego bitplanu	wspólna lista informacyjna dla wszystkich bitplanów
3	bez kompresji	RLC (run length coded)
4	kompresja pozioma	kompresja pionowa
5	short info offsets	long info offsets

W praktyce interesujący nas ANIM5 nie wykorzystuje żadnego spośród tych bitów, lecz można sobie wyobrazić pewne zastosowania, w których mogłyby się one okazać przydatne - przykładowo animacje typu "ping-pong" z użyciem bitu pierwszego, określającego typ danych: set - dane z chunku DLTA zastępują dane na ekranie (w buforze), XOR - na ekran (do bufora) wędruje różnica symetryczna danych z chunku DLTA i bufora.

pad - szesnaście wolnych bajtów z przeznaczeniem do wykorzystania w przyszłości.

Chunk DLTA

w przypadku formatu ANIM5 rozpoczyna się od szesnastu długich słów, wskazujących początek danych dla kolejnych bitplanów. Celem uzyskania adresu bezwzględnego, należy zsumować adres początku chunka w pamięci z zawartością pola odpowiadającego danemu bitplanowi. Jak dotychczas wykorzystuje się pierwsze osiem wskaźników, lecz kto wie co będzie działo się z chwilą wprowadzenia na rynek chipsetu AAA...

Pod uzyskanym w powyższy sposób adresem znajdują się skompresowane dane o zmianach, które zaszły na określonym bitplanie. Mamy tu do czynienia z kompresją pionową, gdzie ekran podzielony jest na podłużne pasy o szerokości jednego bajtu i wysokości równej liczbie linii składających się na obraz. Łatwo policzyć, że ekran o rozdzielczości 320 na 256 pikseli zostanie poszatkowany na czterdzieści kolumn, każda 256 bajtów wysoka. Dane kolumny rozpoczynają się od bajtu określającego ilość segmentów, na które jest ona podzielona. Zero w tym miejscu oznacza, iż w kolumnie nie występują żadne zmiany. Jeżeli jednak zaistniały jakieś różnice, jako kolejny napotykamy bajt informujący o typie segmentu jaki poprzedza. Możliwości są trzy:

1. Bajt zerowy - skompresowana sekwencja powtarzających się wartości. Zaraz za nim znajdziemy ilość powtórzeń, a następnie daną do powielenia.
2. Bajt z ustawionym najstarszym bitem - sekwencja różniących się wartości. Liczba pozostała po zgaszeniu siódmego bitu oznacza ilość danych, które należy przepisać na ekran z miejsca znajdującego się bezpośrednio za omawianym bajtem.
3. Bajt niezerowy ze skasowanym najstarszym bitem - tu się nic nie zmieniło. Należy przesunąć się w dół kolumny o liczbę linii równą temu bajtowi.

We wszystkich przypadkach nie zapominajmy, że jest to kompresja pionowa i aby przesunąć się o daną liczbę linii, do aktualnego adresu należy dodać tę liczbę pomnożoną przez ilość bajtów w linii (w powyższym przykładzie - czterdzieści).

I jeszcze jedna uwaga techniczna: jeśli podczas odtwarzania animacji natkniecie się na chunk CMAP, pamiętajcie by uaktualnić copperlistę zawartymi w nim danymi, które od tej chwili stają się globalne (do czasu napotkania następnego CMAP, oczywiście).

To tyle dobrego (mam nadzieję) na dziś, nasz mini-cykl będzie kontunuowany w następnym numerze gdzie zajmiemy się formatem 8SVX, za dwa miesiące natomiast powrócimy do animacji, tym razem typu ANIM7. Ze względów technicznych (brak miejsca) nie jesteśmy niestety w stanie podarować Wam "animplayera" - za co na wszelki wypadek przepraszamy, jeśli jednak zaistnieje takie zapotrzebowanie, z miłą chęcią umieścimy go na naszym dysku "public domain".

Miłosław "Thorgal" Smyk

SID & VIC - Demonic Labs

ZE SCENY

Było nam bardzo miło otrzymać trzeci egzemplarz wrocławskiego magazynu dyskietkowego SID'n'VIC. Muszę dodać, że śledzę pracę autorów już od pierwszego numeru tego magazynu (dzięki uprzejmości Autorów, którzy dbali o odpowiednio szybkie przysyłanie nowych produkcji). Dlaczego zatem opisujemy dopiero 3 numer? To proste: należało się przekonać co do uporu wydawców i kierunku rozwoju magazynu (mam nadzieję, że Autorzy, czyli Demonic Labs, zrozumieją i wybaczą to niedowiarstwo). Sądzę jednak, że opłacało się poczekać, gdyż dopiero ten numer czyta się "ze smakiem"! A oto kilka szczegółów:

- magazyn SID & VIC jest przeznaczony dla użytkowników Commdore 64 i w nadesłanej do nas postaci (dyskietka) zawierał oprócz "części do czytania" także kilka innych dodatków takich jak proste edytory, mapę do gry, zbiór obrazków grafika z Demonic Labs itp.

- wedle zamysłu twórców, magazyn ten ma być od następnego numeru sprzedawany za jakąś przyzwoitą cenę (pytaj u miejscowego handlarza)

- magazyn jest otwarty na teksty pisane przez jego czytelników - podany jest adres i zasady formatowania tekstów

A co w środku? W środku

znaleźć można opisy, wartościowe(!) felietony, kilka dowcipnych tekstów (m.in. miksowanie znanych z radia i telewizji reklam), ogłoszenia i reklamy.

Na uwagę zasługują zwłaszcza felietony (zwłaszcza ten na temat wyników połączenia handlarzy i Mikrus Copy - brawo Slash!) i dowcipne, "Murphy-podobne" prawa o nieuchronności pecha i błędnej interpretacji.

Na osobne omówienie zasługuje komfort obsługi samego magazynu. Całość "do czytania" wgrywana jest do pamięci w postaci jednego zbioru, który mieści w sobie ponad 100 kB tekstów, dekompresowanych w momencie ich wywołania. Obsługa stron i wybieranie artykułów odbywa się za pomocą rewelacyjnie funkcjonalnego urządzenia zewnętrznego, czyli joystick'a. Na dodatek program wyposażono nawet w możliwość regulowania głośności muzyki (każdy może kręcić czym tylko chce, potencjometrem w monitorze/telewizorze lub programowo - zawsze to jednak fajnie, że jest wybór). Do miłych dodatków należy możliwość resetowania komputera kombinacją klawiszy CTRL i obu SHIFT'ów oraz ponowny reboot dysku (o ile nie mamy włożonego cartridge'a...).

Przy tworzeniu magazynu udzielają się następujące osoby: Slash, Yuppie, Lifter, Raven i Zigzag. Tekstami również zasilili: Chief, Comer, Interlog, Chris.

Korespondencja na adres:

Demonic Labs
57-414 Wrocław 47
skr.poczt. 2314

P.S. Autorów zapraszamy do współpracy z KEBABEM! Jesteście tu mile widziani!

Od dechy do dechy przeczytał i omówił

Paweł Sołtysiński

Opis BootX'a



Era niekontrolowanego panowania wirusów powoli mija. Trudno wymyślić już wirusa, który oszuka stosowane powszechnie zabezpieczenia. Wirusy instalujące się na bootblock'u nie mają przyszłości -każdy może wyświetlić go sobie w postaci ASCII i w razie potrzeby zapisać w jego miejsce jednego z "protectorów". Wirusów plikowych nigdy nie było zbyt wiele, a ponadto gdy ktoś raz już pozna, że przykładowo plik Virus Terminator jest niebezpieczny, to zaraz wszyscy się go bez problemów pozbędą.

Złote czasy wirusów disk-validatorowych skończyły się z chwilą pojawienia się Kickstartu 2.0. W nowszych systemach, disk-validator jest już na stałe w ROM'ie. Przy okazji należy nadmienić, że nasze kochane żyjątka w ogóle są mało odporne na zmiany środowiska (systemu). Są wirusy żyjące tylko przy jednym, określonym Kickstarcie, a w przypadku prób uruchomia go na innym systemie, zamiast harców mikroba, medytuje Guru. Właściwie pozostają na placu boju wirusy linkowe. Tak jak wykrycie wirusa w pamięci nie stwarza poważniejszych problemów, to rozpoznanie z którego pliku się on wykluł jest już sprawą może nie trudniejszą, ale kłopotliwą. Oprócz tego zlikwidowanie podejrzanego bootblock'a czy wirusa plikowego sprowadzało się po prostu do ich usunięcia, natomiast walka z "linkerami" często może kończyć się wylaniem dziecka z kąpielą - wyrzucenie wirusa=strata pliku. Można poradzić sobie z tym poprzez ręczne wycinanie fragmentów kodu z zarażonego programu.

Jak widać wirusy są w defensywie, lecz kto wie czy nie pojawi się ich nowa generacja, która zakpi sobie z dotychczasowych zabezpieczeń? Oby nie...

Dla ułatwienia przy wypędzaniu mikrobów z naszego komputera prezentujemy pełny opis najlepszego chyba programu antywirusowego, czyli BootX'a.

Na wstępie jednak proponuję skrócony kurs wykrywania wirusów.

ALARM:

- przeważnie dowiadujemy się o wirusie z czerwonej migającej ramki, podobnej do Guru Meditation. Najczęściej pojawia się ona w czasie bootowania dyskietki na której nagrany został bootblock testujący wektory pod które wirus mógł się podczepić. Komunikat tego typu nie musi koniecznie znaczyć, że na pewno mamy wirusa, ponieważ podczepione mogą być też inne użytkowe programy, nie wyłączając nawet protektorów - na przykład niech będzie to LVD (Link Virus Detector).

- efektami mogącymi również świadczyć o działalności wirusa są częste uszkodzenia zapisu na dyskietce, bez wiadomej przyczyny. Tu znowu nie mamy do końca pewności czy to na pewno sprawka wirusa, gdyż pamiętajmy, że dyski nie są niezniszczalne, a sprawę mąci jeszcze dodatkowa okoliczność, gdzie błędy z różnych względów mogą wysytąpić nie tylko na uszkodzonej dyskietce.

- innymi przesłankami są "niezidentyfikowane obiekty na dys-

ku", czyli pliki których nie przegrywaliśmy, wolne spacje w startup-sequence (ten objaw jest charakterystyczną cechą kilku wirusów), zmieniona długość plików. Znowu należy zachować zimną krew ze względu na możliwość zakładania plików konfiguracyjnych przez różne programy np. Cygnus nagrywa konfigurację w pliku ceddefaults w katologu s.

Natomiast w przypadku zmiany długości programów to jeżeli wykluczymy operacje osób trzecich jak na przykład dokonanie kompresji, to zmiana ta jest bardzo poważną przesłanką co do obecności intruza i wymaga natychmiastowego sprawdzenia - w końcu pewności w stosunku do naszch podejrzeń nabieramy po komunikacie wirusa, w którym raczy poinformować, że sobie harcuje. Komunikaty te mogą być składane w formie pisemnej, bądź też za pomocą szeregu mniej lub bardziej idiotycznych efekcików, na przykład zamiana pointera myszki na niecenzuralny kształt. Pewności nabieramy również po komunikacie programu antywirusowego, że KONKRETNY wirus został odkryty w pamięci lub na dysku. Ważne jest to czy winowajca został wymieniony po imieniu, gdyż w przeciwnym wypadku obowiązują te same zastrzeżenia jak przy ramkach typu Alert.

PIERWSZA POMOC:

- odsegregować dyski które były w użyciu w okresie pojawienia się podejrzanych okoliczności. Z selekcji mogą zostać wyłączone tylko takie dyski, które przez cały czas były zabezpieczone przed nagraniem i co do których istnieje pewność, że nie są źródłem kłopotów.

- wyczyścić sumiennie pamięć - najlepiej wyłączając komputer z sieci.

- ze 100% niezarażonego dysku uruchomić wybrany program antywirusowy. Ze względu na to, że w niniejszym artykule koncentrujemy się na programie BootX, będziemy więc opisywać postępowanie jakby tym wybranym był właśnie on.

WIELKIE POLOWANIE:

- najpierw wyszukujemy wirusów bootblokowych. Wkładamy po kolei poszczególne dyski i bacznie obserwujemy jak zachowa się BootX. Wyświetlenie nazwy bootbloka powoduje jego rozpoznanie i pozwala na trafną ocenę. Jeżeli pojawi się nazwa wirusa, należy skorzystać z dobrodziejstwa funkcji Install - patrz opis. Gdy BootX nie rozpozna boota, przyglądamy się jego zawartości przedstawionej w postaci tekstu ASCII. Czasem autor wirusa zostawia tam wszystkie informacje. Gdy nic podejrzanego nie zostanie znalezone, BootX nie jest w stanie już nic poradzić. Jego główną (chyba jedyną) wadą jest brak opcji analizy bootbloku jak to ma miejsce w Virus Expercie. Wyjątkowo sięgamy po tego drugiego i jeżeli nadal nic z tego nie wynika a my dalej jesteśmy pełni podejrzeń, pozostaje resourcer i wnikliwa analiza kodu boota.

- BootX sprawdza wirusy plikowe i linkowe za jednym zamachem. Służy do tego opcja Check Files. W tym momencie nasze działanie ogranicza się do wydawania wyroku na zidentyfikowane mikroby. Nie zawsze jest możliwe znalezienio go, przykładowo jeżeli program do którego się dołączył został następnie skompresowany programem innym niż Power Packer. W stosunku do szczególnie podejrzanych plików możemy przeprowadzić proces dekompresji, by następnie powtórzyć procedurę sprawdzającą. Gdy nic nie zostanie znalezione, a nasza chorobliwa podejrzliwość nie została zaspokojona, zawsze pod ręką jest resourcer.

- znalezione wirusy bezwzględnie usuwamy, chyba że są one nowe i chcemy je włączyć do biblioteki brainfile, zawierającej zapamiętane booty.

NAPRAWIANIE SZKÓD:

- jeżeli znaleźliśmy Saddama, możemy skorygować uszkodzone przez jego wygłupy bloki. Służy do tego opcja Repair Disk.

- w przypadku uszkodzenia plików typu biblioteki, fonty itp. uzupełniamy je takimi samymi z innych dysków. Gdy wirus nagrzebał coś na dysku i w innych plikach, pracę zlecamy najlepiej FixDiskowi, choć oczywiście nie wszystkie szkody dadzą się naprawić. Pliki uratowane przegrywamy na inny dysk, a stary formatujemy.

CZYNNOŚCi TOWARZYSZĄCE:

- do tej kategorii zaliczają się działania nie wynikające z potrzeby lecz z nakazów moralności i etyki. Warto więc powiadomić o naszych kłopotach kolegów którzy pożyczyli nam ostatnio dyski i którym my pożyczaliśmy. Niech sami sprawdzą czy nie mają wirusa i nie jest tu ważne kto go niechcący rozpowszechnił.

- pamiętajmy o szerokiej profilaktyce (ojej brzmi to jak z ulotki o AIDS!)

Tak wygląda przeważnie przebieg działania z wirusami. Teraz obiecany opis.

Funkcje dostępne z ekranu

QUIT : Wyjście z BootX'a

READ BB: Wgrywa bootblock

z wybranej stacji do pamięci.

WRITE BB : Instaluje bootblock z pamięci na dysk w wybranej stacji.

DF0: : Wybiera stację DF0: do pracy z programem.

DF1: ... DF3: : Analogicznie jak DF0:.

LOCK DRIVE: Zabezpiecza system (przy wkładaniu dysku do stacji) przed wirusami lokującymi się w disk-validatorze. Użyj tej funkcji przy sprawdzaniu lub naprawianiu dysku np. z wirusem Saddam.

INSTALL : Instaluje wybrany bootblock na dysk w danej stacji. Wyboru danego bootblock'u dokonuje się spośród wielu bootblock'ów "wbudowanych" w program (jak tworzyć własny zestaw bootblocków do instalacji pokazano w Kebabie 9`92).

< > : Wybór poprzedniego/następnego wbudowanego bootblock'a.

To były wszystkie funkcje dostępne bezpośrednio z ekranu. Teraz kolej na opisanie menu.

Menu PROJECT

LOAD BOOTBLOCK LIBRARY Uruchamiając tę opcję można załadować inną od standardowej bibliotekę bootblock'ów. Standardową biblioteką jest *BootX.BBlib*, ładowana automatycznie przy uruchamianiu programu.

LOAD RECOG FILE : BootX do rozpoznawania wirusów i bootblock'ów używa danych z pliku *BootX.Recog* i właśnie ta opcja pozwala na wgranie alternatywnego do oryginalnego i ładowanego przy starcie programu, pliku Recog.

Uwaga! Jeżeli posiadasz Kickstart 1.3 i smutno Ci, że nie możesz używać BootX'a w jego ulepszonych wersjach, to śpieszymy donieść, że nowe pliki recog pasują do starszych wersji. Nie wiadomo tylko, czy wraz z czasem, autor Peter Stuer, nie zmieni ich formatu...

SHOW KNOWN BOOTBLOCKS : Wyświetla listę znanych programowi bootblock'ów.

SHOW KNOWN BOOTVIRUSES : Wyświetla listę wirusów bootblock'owych. Na liście tej uwzględniany będzie tylko jeden wirus z rodziny swych klonów, np. Lamer Exterminator wypisany zostanie tylko raz, ale jego mutacje są znane.

SHOW KNOWN FILEVIRUSES Wyświetla spis znanych programowi wirusów plikowych.

GO TO SLEEP : "Usypia" program, tworząc małe okienko BootX. Powrót do programu następuje po "kliknięciu" na nim kolejno lewym i prawym przyciskiem myszy.

ABOUT : Wyświetla informacje o programie BootX.

Menu BOOTBLOCKS

LOAD...: Pozwala na wybranie pliku i wczytanie go jako bootblock'a w celu późniejszego zainstalowania. Program ostrzega, czy plik jest poprawnym bootblock'iem AmigaDOS'u. Jeśli jego suma kontrolna nie będzie się zgadzała, to BootX może ją ewentualnie poprawić.

SAVE...: Umożliwia nagranie bootblock'a z pamięci na dysk w postaci pliku. W ten sposób można stworzyć cały katalog bootblock'ów w celu ich późniejszego instalowania, wymiany etc.

CLEAR BRAINFILE : Usuwa z pamięci bibliotekę brainfile. Program najpierw prosi o potwierdzenie tej decyzji.

LOAD BRAINFILE : Umożliwia wgranie do pamięci pliku rozszerzającego bibliotekę znanych wirusów, tzw. brainfile.

SAVE BRAINFILE : Nagrywa z pamięci plik brainfile na dysk.

VIEW BRAINFILE : Pokazuje zawartość biblioteki brainfile.

MERGE BRAINFILE : łączy plik typu brainfile z dysku z biblioteką brainfile w pamięci. W ten sposób można zwiększyć liczbę znanych bootblock'ów. Powtarzające się dane będą zignorowane.

LEARN : Zapamiętuje znajdujący się w buforze bootblock przez dodanie go do biblioteki brainfile w pamięci.

Menu FILES

CHECK FILES : Wyświetla okno do zmiany opcji używanych do szukania wirusów linkowych. Wybranie funkcji 'Check files' rozpoczyna poszukiwania.

SHOW REPORT AGAIN : Wyświetla ostatnio wygenerowany raport o poszukiwanych wirusach.

SAVE REPORT : Nagrywa ostatnio wygenerowany raport na dysk, jako plik tekstowy.

CHECK DISK : Sprawdza dokładnie cały dysk: bootblock, disk-validator, pliki i naprawia uszkodzone sektory.

REPAIR DISK : Czyta dysk szukając sektorów uszkodzonych przez wirusy. Jeśli uszkodzenia będą pochodziły od Saddama, zostaną naprawione, jeśli od Lamer Exterminator'a - zostaną zaznaczone.

Menu MISCELLANEOUS

CHECK MEMORY : Szuka w pamięci wirusa oraz wyświetla struktury mogące się przez niego zmienić. Jeśli BootX rozpozna rezydentny program zmieniający wektory, to wypisze jego nazwę. W przeciwnym wypadku program odznaczy to komunikatem "Please check". Uwaga! Komunikat ten nie oznacza od razu wirusa.

RESET VECTORS : Resetuje pięć najważniejszych dla wirusów wektorów. Znacznie bezpieczniej jest jednak wyłączyć

Amigę z sieci, jeśli wirus jest w pamięci.

INSTALL RESIDENT LVD : Umieszcza w pamięci LVD program zabezpieczający przed wirusami linkowymi. Jest on aktywny w pamięci nawet po resecie.

Menu SETTINGS

DETECT DISKCHANGE : Wyłączenie tej opcji blokuje rozpoznawanie, czy dysk został zmieniony. Umożliwia to łatwe instalowanie bootblock'u z pamięci na dysk, szczególnie dla posiadaczy jednej stacji dysków.

SHOW HELP : Włącza/wyłącza tryb HELP. Przy włączonym trybie HELP można wybrać dowolną opcję z menu lub okna (ekranu) i otrzymać w ten sposób jej opis. W niniejszym artykule wykorzystano przetłumaczony tekst z tegoż "helpa" (wersja BootX'a 4.50 PL ma standardowo polskie opisy).

PAUSE AFTER PAGE : Po wybraniu tej opcji wszelkie długie informacje będą wyświetlane strona po stronie.

CHECK MEMORY AT STARTUP : Decyduje czy BootX będzie testował pamięć przy każdym uruchomieniu programu.

CHECK DISK-VALIDATOR : Po wybraniu tej opcji BootX będzie sprawdzał disk-validator'a po każdej zmianie dysku.

AUDIBLE ALARM : Włączenie tej opcji powoduje odtwarzanie sampla z pliku *BootX.alarm*, w przypadku odczytania z dysku nieznanego bootblock'a.

SET COLORS : Umożliwia zmianę kolorów używanych na ekranie programu.

LOAD SETTINGS... : Ładuje konfigurację programu.

SAVE SETTINGS... : Nagrywa konfigurację BootX'a pod nazwą *BootX.prefs*.

SAVE SETTINGS AS... : Nagrywa konfigurację na dysk, ale pod inną, wybraną przez nas nazwą.

To już wszystkie funkcje BootX'a. Pamiętajmy jednak, że żadnemu programowi nie należy bezgranicznie wierzyć. Nie chodzi mi w tym miejscu oczywiście o próby oszukania użytkownika przez program, lecz o działanie niektórych wirusów. Otóż istnieją żyjątka modyfikujące system w ten sposób, że każda próba odczytu bootblock'a, spowoduje wyświetlenie standardowego boota AmigaDos, chociaż na bootbloku biwakuje sobie w najlepsze wirus. Aby tego uniknąć, pamiętajmy, że przed rozpoczęciem poszukiwań, należy bezwzględnie wyczyścić pamięć. Najlepiej wyłączyć Amigę z sieci.

Zbigniew "Zybul" Piotrowicz

Amos VIII

SOFTWARE

AMOŚ - czyli AMOS po polsku

W ósmej już części kursu oprócz dalszych opisów komend, spróbujemy poradzić sobie z często powracającym w Waszych listach problemem używania polskich znaków

Wielu użytkowników Amosa nęka nieumiejętność zastosowania polskich czcionek we własnym programie. Możliwym wyjściem z sytuacji jest zmiana fonta (komendą Setfont) na dowolny, zawierający potrzebne nam litery. Zmieniony w ten sposób alfabet, będzie działał jedynie w trybie tekstu graficznego. Należy zastąpić więc wszystkie rozkazy Print wykorzystujące nasze narodowe znaki, na rozkazy Text.

Powyższy sposób pociąga za sobą wiele niedogodności. W niniejszym artykule przedstawimy drugi, bardziej elegancki i wygodniejszy. Jego niepodważalną zaletą będzie praca w trybie tekstowym, a ponadto brak konieczności wprowadzania jakichkolwiek rozkazów startujących bezpośrednio w naszym programie (!), zarówno w edytorze jak i po skompilowaniu.

Zanim jednak nam się wszystko uda, musimy przebrnąć przez kilka problemów. Pierwszym z nich jest wybór tzw. "standardu" polskich liter. Dlaczego słowo "standardu" napisaliśmy w cudzysłowiu?

Otóż niektórzy amigowscy profesjonaliści zazdroszczą ibeemowskim profesjonalistom tego, że co nowy program to nowy standard np. zapisu grafiki (obrazków) albo też nowy "standard" polskich liter. Bóg jeden wie ile w końcu "standardów" polskich liter powstało na pecetach. Na pewno da się tą liczbę określić jednym słowem: wiele...

Po prostu każdy "profesjonalista" chce się jakoś wykazać i ma nadzieję, że właśnie on będzie tym najmądrzejszym i zapisze złote karty w historii polskiej komputeryzacji. Efekty każdy widzi. Podobną niestety tendencję da się ostatnio zauważyć wśród amigowskich profesjonalistów. Powstający z tego powodu chaos i zamieszanie nie wiadomo komu może służyć ale na pewno nie Amidze...

W związku z powyższym oraz ponieważ nie chcielibyśmy zostać kiedykolwiek posądzeni o "profesjonalizm", postanowiliśmy przyjąć, że kto pierwszy, ten lepszy. Dlatego też oddajemy prawo do zapisania złotych kart ks. Pikulowi, który (chyba niechcący) stał się głównym autorytetem w sprawie polskich liter na Amidze.

Nasz amosowski sposób będzie się zatem ściśle opierał na tym co zrobił kiedyś (jako pierwszy na dużą skalę) ks. Pikul. Ostatnio promowany jest przez niego (pod wpływem profesjonalistów) nowy i zmieniony format, więc jeżeli zdecydujecie się na nowsze obłożenie to proszę bardzo. Obok tabelki ze starymi wartościami podajemy nowy standard. Małe litery uzyskiwane są przy wciśnięciu klawisza Alt i litery podstawowej np. Alt+a=ą (wyjątkiem jest litera ź - Alt + x), zaś duże otrzymujemy podobnie, ale dodatkowo wciskamy Shift. Standardowe obłożenie klawiatury w edytorze Amosa nie uwzględnia możliwości wprowadzania znaków różnych (z drobnymi wyjątkami), od tych widocznych na klawiszach. Z tego względu niezbędna jest zmiana nie tylko fonta, ale i całego "keymapu". Dokonujemy tego programem P.J. Hickman'a *Keyboard_Definer.AMOS*, znajdującym się na dyskietce Amos Extras (nie mylić z "Ekstrasami" Workbench`a). Edytujemy nim obłożenie klawiatury poprzez przyporządkowanie odpowiedniemu klawiszowi, stosownej watości Ascii (patrz tabelka, kolumna Dec). Obsługa Definer'a jest znacznie prostsza od popularnego SetKey'a, więc nie będziemy mieli z nim najmniejszych problemów. Posiadaczom "angielskiej" klawiatury proponuję modyfikację obłożenia na bazie pliku `American.key` lub `English.key`.

Należy zwrócić uwagę na rodzaj posiadanej klawiatury, gdyż wersja amerykańska różni się nieco od angielskiej, a w sklepach zwyczajowo już używa się w stosunku do obydwu określenia 'angielskie'. Posiadacze klawiatury niemieckiej, nie mają właściwie wyboru, muszą zmieniać plik `German.key`. Po skończonej edycji nagrywamy obłożenie na dysk.

Kolejną sprawą jest narysowanie fonta z polskimi literami. Najłatwiejszym jest wykorzystanie programu *FED* z dysku Extras (tym razem tego dołączanego do Workbench'a 1.3). Rysujemy dodatkowe litery, pamiętając o ich stosownym miejscu (patrz tabelka, kolumna Hex), a następnie nagrywamy nasze dzieło na dysk. Teraz za pomocą konwertera *Font_Convert.AMOS*, przekształcamy naszego fonta, na format rozpoznawany przez Amosa. Jeżeli wybierzecie sobie starszą wersję standardu ks. Pikula, to nie trzeba męczyć się z rysowaniem od podstaw, bowiem istnieje bardzo wiele fontów spolszczonych stosownie do wymogów tegoż formatu.

Ostatnim zadaniem jest zamiana oryginalnych plików z katalogu AMOS_System : `default.key` i `default.font`, na nasze uprzednio przygotowane odpowiedniki. Po tym drobnym zabiegu możemy używać już polskich znaków w tekstowym trybie Amosa.

Na dysku Kebab Public Domain #3 umieszczone zostały gotowe amosowskie pliki .font i .key do wykorzystania ich w instalacji polskich liter.

"Stary" standard ks.Pikula

litera	ASCII		litera	ASCII	
	Hex	Dec		Hex	Dec
ą	$E6	230	Ą	$C6	198
ć	$E7	231	Ć	$C7	199
ę	$EA	234	Ę	$CA	202
ł	$EE	238	Ł	$CE	206
ń	$F1	241	Ń	$D1	209
ó	$F3	243	Ó	$D3	211
ś	$F5	245	Ś	$D5	213
ź	$FB	251	Ź	$DB	219
ż	$FE	254	Ż	$DE	222

Po zeszłomiesięcznej przerwie wracamy do poznawania Amosa. W tym odcinku naszego kursu poznamy funkcje i operacje arytmetyczne. W operacjach tych możemy używać nie tylko podstawowych działań takich jak dodawanie, odejmowanie, mnożenie, dzielenie, potęgowanie i pierwiastkowanie. Dostępne są rozmaite funkcje, operacje logiczne And, Or i Not, Do tych obliczeń najczęściej używa się zmiennych rzeczywistych, gdyż wynikami funkcji na przykład trygonometrycznych są takie właśnie wartości. Aby stosownie wykorzystać oferowane nam możliwości, nie wolno zapominać o dziedzinach odpowiednich funkcji, założeniach i pułapkach typu 1/A#, gdzie A#=0. Ponadto w katalogu libs na dysku musi znajdować się biblioteka matematyczna. Ja zakładam, że w szkole poznaliście już trochę ten dział matematyki, więc opisu co to jest np. logarytm, nie będę podawał.

`Sin(A#)` - funkcja sinus.

`Cos(A#)` - funkcja cosinus.

`Tan(A#)` - funkcja tangens.

Funkcji Cotangens Amos nie posiada, lecz obliczamy go następująco:

```
CTG#=1/Tan(A#)
```

`Acos(A#)` - arcus cosinus.

Nowa wersja

litera	ASCII		litera	ASCII	
	Hex	Dec		Hex	Dec
ą	$E2	226	Ą	$C2	194
ć	$EA	234	Ć	$CA	202
ę	$EB	235	Ę	$CB	203
ł	$EE	238	Ł	$CE	206
ń	$EF	239	Ń	$CF	207
ó	$F3	243	Ó	$D3	211
ś	$F4	244	Ś	$D4	212
ź	$FA	250	Ź	$DA	218
ż	$FB	251	Ż	$DB	219

Arcus sinus obliczamy przykładowo tak:

```
ARCSIN#=Pi#/2.0-Acos(A#)
```

`Pi#` jest w Amosie stałą.

`Atan(A#)` - arcus tangens.

`Hsin(A#)`, `Hcos(A#)` i `Htan(A#)` to funkcje hiperboliczne.

`Degree` - powoduje, że argumenty w powyższych funkcjach będą traktowane jako stopnie.

`Radian` - powoduje, że argumenty będą interpretowane jako dane w mierze łukowej, czyli radiany.

`Ln(A#)` - logarytm naturalny (o podstawie e).

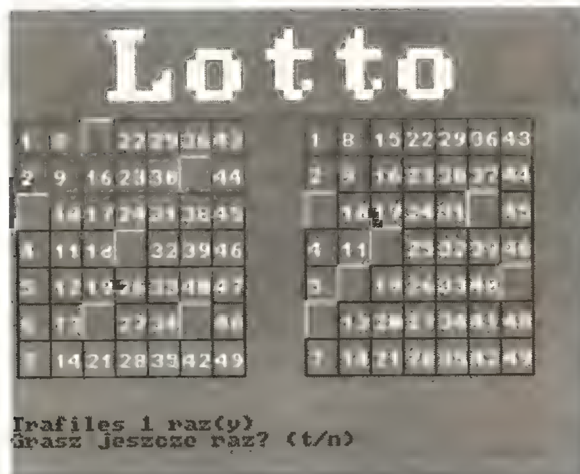
`Log(A#)` - logarytm dziesiętny.

`Exp(A#)` - funkcja wykładnicza e^A#.

`Sqr(A#)` - pierwiastek kwadratowy.

`Abs(A#)` - wartość bezwzględna.

`Int(A#)` - część całkowita. Nie mylić z zaokrąglaniem, gdyż wartość Int powstaje nie po "wyrównaniu" do najbliższej liczby całkowitej, lecz po zmniejszeniu liczby do całości. Tak więc Int(99.99)=99, a Int(-99.99)=-100.



`Sgn(A#)` - funkcja signum.

Amos pozwala nie tylko na korzystanie ze standardowych funkcji, ale mamy również możliwość tworzenia naszych własnych. Niezbędna do tego jest deklaracja funkcji użytkownika:

```
Def Fn NAZWA(parametry)=nasza_funkcja
```

Na przykład stworzymy funkcję f(x)=(sin x1 + cos x2) * sgn (X1+X2)

```
Def Fn MOJA(X1#,X2#)=(Sin(X1#)+
          Cos(X2#))*Sgn(X1#+X2#)
```

Następnie do woli możemy jej używać np.

```
Print Fn MOJA(5.0,5.1)
```

Uwaga : Rozkaz deklaracji funkji musi znajdować się sam w jednej linii i funkcja deklarowana w jednej procedurze nie będzie dostępna w innej.

`Fix(N)` - rozkaz ustalający tryb wyświetlania zmiennych rzeczywistych. W zależności od parametru wyświetlane będzie N miejsc po przecinku, a gdy parametr będzie ujemny, liczby wyświetlane będą w postaci wykładniczej.

Teraz trochę pomocnych operacji...

`Max(X1,X2)` - funkcja przyjmująca wartość większego z dwu podanych argumentów.

`Max(X1$,X2$)` - funkcja przyjmująca w wyniku zmienną tekstową rozpoczynającą się znakiem o większym kodzie ASCII. Gdy pierwsze litery są takie same, decyduje wtedy druga, lub ewentualnie dalsze.

`Min(X1,X2)` - podobnie jak Max(X1,X2), ale przyjmuje wartość mniejszą.

`Min(X1$,X2$)` - analogicznie do Max(X1$,X2$), ale "wygrywa" zmienna o mniejszym kodzie.

`Add A,B,MIN To MAX` - Dodaje do zmiennej A zmienną B, lecz gdy wynik wyjdzie poza granicę określoną przez maksimum, to przyjmuje jego wartość. Identycznie sprawa ma się z minimum.

`Swap(A,B)` - zamienia ze sobą wartości zmiennych lub ciągi znaków.

W tym miejscu chciałbym obalić pewien mit krążący wśród programistów języków wysokiego poziomu. Wielu śmiało się z tezy mego kolegi, że aby zamienić wartości dwu zmiennych, niekoniecznie należy wykorzystać zmienną pomocniczą np.X i wykonywać operację posługując się wzorcem X=A : A=B : B=X. Otóż jest inny sposób nie wykorzystujący zmiennej pomocniczej (nie powiem że słuszniejszy, aczkolwiek istnieje).

Proszę sobie sprawdzić samemu: A=A+B : B=A-B : A=A-B. Po tej małej dygresji wracamy do Amosa.

Oprócz wyliczanych wartości z rozmaitych funkcji, czasem przydać się mogą liczby losowe (dla fanatyków ścisłych określeń - liczby pseudolosowe).

`Rnd(N)` - funkcja losująca liczbę całkowitą z przedziału od zera do N.

`Randomize N` - ustawia parametr początkowy do procedury losowania.

`Randomize Timer` - ustawia ten parametr stosownie do czasu, który upłynął od włączenia komputera. Użycie tego rozkazu zwiększa przypadkowość.

No i część arytmetyczną mamy już za sobą. W listingach znajdziecie program zawierający przykłady na liczenie funkcji, w najkrótszy sposób obrazujących takie zagadnienia jak tablicowanie wyników (aby płynnie a-

nimować później rozmaite obiekty, czasowo opłaca się bardziej przeliczyć wcześniej ich powiedzmy sinusoidalną trasę, niż dokonywać kalkulacji na bieżąco), wykorzystywanie tych danych, rysowanie funkcji. Ponadto macie przed sobą okazję darmowej gry w Totolotka (czy w Lotto jak to się z nowoczesna mówi), dzięki której nie tylko utrwalicie sobie Amosa, ale i przypomnicie sobie jakie ważne są elementy rachunku prawdopodobieństwa - liczyliście w szkole prawdopodobieństwo trafienia szóstki?

W programie Lotto na pewno zwrócicie uwagę, że grafika jak na te kilka instrukcji, wcale nie jest taka zła. W listingu nie ma możliwości umieszczenia obrazków, zmuszony więc byłem do użycia iście partyzanckich metod w celu zmuszenia programu do ładniejszego wyglądu. Wy zaopatrzeni w takie kolosy jak Dpaint czy Real 3D, stworzycie sobie do swych amosowskich procedur grafikę z prawdziwego zdarzenia, a tymczasem przeanalizujcie oba listingi, bo z pewnością operacje tam przeprowadzone jeszcze nie raz okażą się niezbędne w innych, nawet zupełnie odmiennych programach.

Zbigniew "Zybul" Piotrowicz

JAK ZROBIĆ WŁASNY TURBOLOADER czyli programowanie stacji 1541/71 część 5



Poprzedni odcinek zakończyczyliśmy krótkim programem odczytującym z dysku sektor 18,0. Skorzystaliśmy przy tym z usług DOS-u. Już na pierwszy rzut oka widać, że mechanizm ten może być czasem bardzo użyteczny. Nie zawsze przecież musimy robić coś (czytać, pisać, itp) z dysku z kosmiczną prędkością. Jeżeli potrzebujemy dajmy na to odczytać trzy sektory i pozostawić je w RAM stacji (czy to jako program, czy w innym celu) i jeżeli musimy robić to powiedzmy raz na 10 minut, to naprawdę nie ma dla nas znaczenia, czy stacja zrealizuje to w ciągu 0.6 czy 0.2 sekundy. Poza tym, jak nietrudno się domyśleć, programowanie stacji to ciągłe kompromisy pomiędzy szybkością, a długością programu. Przeważnie tak się zdarza, że jesteśmy zmuszeni spowolnić działanie programu, aby w ogóle zmieścił się on nam w stacji.

Bardzo pomocny w takiej sytuacji staje się Dyskowy System Operacyjny, o którym powiedziałem już kiedyś tyle złego. Na szczęście dla nas, zdecydowana większość błędów zdarzyła się programistom przy pisaniu części zarządzającej plikami, katalogiem i BAM-em dysku. Nie ma się co temu dziwić, gdyż procedury te modyfikowane były za każdym razem, gdy pojawiał się nowy model stacji Commodore. Taka jest prawda. Nie wiadomo czym kierowała się firma Commodore nie chcąc się zdecydować na napisanie porządnego DOS-u zaczynając od zera dla każdej nowej stacji. Czyżby kompatybilność stacji do komputera PET (poprzednika C-64) ze stacją 1541, czy 1571 była taka ważna, pomimo że PET i C-64 zgodne były praktycznie tylko przez BASIC i trochę podobny ROM? Wszystko to powodowało, że błędów było coraz więcej. Nie ma się więc co dziwić, gdy w ROM-ie 1541, czy 1571 znajdzie się fragmenty procedur, które nigdy nie są do niczego używane, a są pozostałościami po poprzednich modelach.

Jak już powiedziałem, na szczęście dla nas, procedury, których będziemy używać są na tyle krótkie i proste, że łatwo jest prześledzić co faktycznie one robią. Daje nam to gwarancję, że nasz program będzie "chodził" tak, jak sobie tego życzymy. Powróćmy jednak do naszego przykładu z poprzedniego odcinka i wyjaśnijmy jak to wszystko działa.

Na początku ustawiamy komórki $06 i $07 na wartości odpowiednio ścieżki i sektora, który mamy zamiar odczytać, po czym wpisujemy wartość $80 do komórki $00. I w tym miejscu mała uwaga. Wszyscy przyzwyczajeni są do tego, że nie można używać komórek $00 i $01 gdyż są to rejestry procesora sterujące wbudowanym układem wejścia/ wyjścia. W C-64/128 rejestr ten jest używany do zmian konfiguracji pamięci, oraz do obsługi mag-

netofonu. W stacji natomiast $00 i $01 to normalne komórki strony zerowej, nie różniące się niczym od pozostałych. Kontynuujmy jednak analizowanie programu.

Po ustawieniu komórki $00 program czeka już tylko, aż zgaśnie jej najstarszy bit, co oznacza, że operacja została wykonana. Tak, ale mógł przecież wystąpić jakiś błąd. Czy da się to jakoś sprawdzić? Nic prostszego! Wystarczy odczytać zawartość komórki $00. Cztery najmłodsze bity oznaczają właśnie numer błędu.

A oto pełna ich lista, zaczerpnięta z instrukcji obsługi stacji 1571:

$00,$01 - O.K.
$02 - SECTOR NOT FOUND
$03 - NO SYNC
$04 - DATA BLOCK NOT FOUND
$05 - DATA BLOCK CHECKSUM ERROR
$06 - FORMAT ERROR
$07 - VERIFY ERROR
$08 - WRITE PROTECT ERROR
$09 - HEADER BLOCK CHECKSUM ERROR
$0A - DATA EXTENDS INTO NEXT BLOCK
$0B - DISK ID MISMATCH
$0E - SYNTAX ERROR
$0F - NO DRIVE PRESENT

Omówimy teraz dokładniej te błędy, które najbardziej będą nas interesowały. A więc po kolei:

$00, $01 - Wartości te w jednakowym stopniu określają, że ostatnia operacja zakończyła się pomyślnie. Należy zawsze przyjmować, że gdy wartość bajtu statusowego jest mniejsza od dwóch, to wynik jest O.K.

$02 - SECTOR NOT FOUND - Błąd ten pojawia się, gdy kontroler nie może odnaleźć na konkretnej ścieżce nagłówka sektora, który chcemy odczytać. Może to świadczyć o nieprawidłowym numerze sektora (na przykład chcemy odczytać sektor 20 ze ścieżki 18, pomimo że jest na niej faktycznie zapisanych tylko 19 sektorów (o numerach od 0 do 18)). Częściej świadczy on jednak o uszkodzeniu samego nagłówka. Tak zepsutego sektora NIE DA się odzyskać korzystając z usług systemu operacyjnego. Zwykle jednak wystarcza napisanie własnego krótkiego programu szukającego poprzedniego sektora, ignorującego odpowiednią ilość znaków synchronizacji i odczytującego interesujący nas blok danych (oczywiście, jeżeli i ten nie jest uszkodzony).

$03 - NO SYNC - Pojawia się, gdy kontroler nie jest w stanie odnaleźć na żądanej ścieżce ani jednego znaku synchronizacji. (Przypomnijmy, że znak ten sygnalizuje początek każdego nagłówka sektora i każdego bloku danych.) Błąd ten pojawia się, gdy na przykład nie włożymy dysku do stacji, czy też chcemy coś odczytać ze ścieżki, która nie została nigdy sformatowana. Może też się czasem zdarzyć, że otrzymamy NO SYNC, gdy głowica nie jest dokładnie ustawiona nad zapisanym śladem magnetycznym ścieżki. Należałoby w takim wypadku próbować przesuwać głowicę o pół ścieżki w jedną, czy w drugą stronę i dopiero, gdy to nie da rezultatu, przyjąć że ścieżka jest uszkodzona. Zazwyczaj jednak, pomimo niedokładnego ustawienia głowicy znaki synchronizacji są znajdowane (może nie wszystkie, ale zawsze).

$04 - DATA BLOCK NOT FOUND - Oznacza, że kontroler znalazł żądany sektor, ale bloku danych albo w ogóle nie było, albo miał uszkodzony znak synchronizacji (czyli też można przyjąć, że go nie ma, bo i tak nie damy rady go odczytać). Ogólnie można powiedzieć, że błąd ten jest generowany jeżeli wartość znacznika bloku danych odczytanego sektora różni się od zawartości komórki $47. (Standardowo, aby odróżnić nagłówki od bloków danych bezpośrednio po znaku synchronizacji zapisywane są znaczniki: nagłówka ($08) albo bloku danych ($07)). W takim wypadku można przyjąć, że błąd ten ma charakter wyłącznie informacyjny, gdyż pomimo jego wygenerowania odpowiedni sektor i tak jest załadowany tam, gdzie chcemy (pod warunkiem oczywiście, że jedyną niepoprawnością w bloku danych jest inny znacznik). Gdy bloku danych nie ma, zamiast sektora załadowane zostanie to, co znajduje się po najbliższym znaku synchronizacji. (Na przykład nagłówek następnego sektora i inne śmieci.) Zazwyczaj nie mamy do czynienia z zabezpieczonymi w ten sposób dyskami i komunikat ten musimy traktować jako błąd (w dodatku dość trudny do naprawy).

$05 - DATA BLOCK CHECKSUM ERROR - Informuje, że suma kontrolna bloku danych odczytana z dysku nie zgadza się z sumą wyliczoną poprzez prze-EOR-owanie wszystkich bajtów sektora. Rownież może zdarzyć się, że jest to zrobione celowo, ale nie jest to częsty przypadek. Zazwyczaj błąd ten świadczy o tym, że w którymś miejscu odczytywanego bloku danych kontroler "zgubił" jakieś bity (np. z powodu wad nośnika), co spowodowało nieprawidłowe zinterpretowanie informacji. Po wystąpieniu tego błędu sektor znajduje się wprawdzie w pamięci RAM, ale można przyjąć, że od miejsca wystąpienia błędu (a nie wiemy dokładnie gdzie to nastąpiło) do końca sektora wszystkie bajty są błędne. (Gdy kontroler źle odczytał jakieś bity wszystko mogło się poprzesuwać, po czym było jeszcze dekodowane, w rezultacie czego mamy nie obrazujący niczego ciąg bajtów.) Błąd ten często daje się naprawić (szczególnie, gdy nasz dysk nie jest uszkodzony, a tylko ma "niepewne" miejsca) poprzez kilkakrotne powtórzenie odczytu (aż do otrzymania komunikatu O.K.) i ponowne zapisanie sektora na dysk.

$07 - VERIFY ERROR - Oznacza, że w czasie weryfikacji sektora wykryto różnice między tym, co na dysku, a tym co w pamięci RAM. Należy w takim wypadku

spróbować ponownie zapisać sektor, a gdy i to nie da rezultatu, można przyjąć, że dysk jest uszkodzony.

$08 - WRITE PROTECT ERROR - Błąd ten jest generowany, gdy próbujemy zapisać sektor na dysk, który ma zaklejone wycięcie ochrony przed zapisem. Przeglądając ROM stacji można znaleźć miejsca, gdzie stan bariery świetlnej ochrony przed zapisem jest testowany i ewentualnie zwracany jest błąd $08. Nie trzeba się jednak obawiać, że jakiś program coś nam zapisze na "zaklejonym" dysku, gdyż nawet jeśli zignorowałby on stan bariery to i tak elektronika stacji nie dopuści, aby głowica choć przez chwilę przełączona została na zapis. W praktyce wygląda to tak, że zapisujemy sektor, czy nawet formatujemy ścieżkę, wszystko wygląda normalnie, a i tak na dysku zostaje to co było wcześniej.

$09 - HEADER BLOCK CHECKSUM ERROR - Niezgodność odczytanej i obliczonej sumy kontrolnej nagłówka. W rezultacie sektor nie jest załadowany, gdyż nie ma pewności, że jest to właśnie ten sektor o który nam chodzi. Błędy mogły się przecież pojawić w bitach opisujących właśnie numer tego sektora. Jeżeli błąd ten powstał przez nieodpowiedni zapis nagłówka, czy też w wyniku mechanicznego uszkodzenia dysku, to o ile nie jest uszkodzony sam blok danych powinno dać się go odczytać podobnym sposobem, jaki opisywałem przy błędzie $02.

$0A - DATA EXTENDS INTO NEXT BLOCK - Kontroler próbuje zawsze znaleźć znak synchronizacji następnego nagłówka zaraz po zapisie bloku danych. Jeżeli nie pojawi się on w odpowiednim czasie, generowany jest błąd $0A. Na dobrych, poprawnie sformatowanych dyskach błąd ten jednak nie występuje, chyba że zapisujemy sektor z prędkością wolniejszą niż przyjęta na danej grupie ścieżek. Jeżeli korzystamy z usług DOS-u, nie musimy się tym martwić, gdyż system do tego nigdy nie dopuści. Problem zbyt długiego bloku danych może też zaistnieć, gdy próbujemy zapisać dysk obracający się wewnątrz koperty z wyraźnymi oporami, co może spowodować zwolnienie jego obrotów i w rezultacie zmniejszenie prędkości zapisu.

$0B - DISK ID MISMATCH - Występuje, jeżeli zażądaliśmy odczytu, czy zapisu na dyskietkę, która nie została wcześniej zainicjowana. Może się też pojawić, gdy interesujący nas sektor ma w nagłówku inne znaki ID niż reszta dysku. (DOS nie dopuszcza takiej sytuacji, aczkolwiek fizycznie jest to możliwe do zapisania i może stanowić zabezpieczenie dysku przed kopiowaniem programami kopiującymi, z wyjątkiem NIBBLER-ów.)

$0F - DRIVE NOT PRESENT - Błąd ten generowany jest za każdym razem, gdy odwołujemy się do napędu numer 1, który jest dostępny wyłącznie w stacjach dwunapędowych. Dla stacji 1541/71 jako numer napędu trzeba zawsze podawać wartość 0.

To tyle opisu błędów. Zajmiemy się teraz następną sprawą, czyli powiemy CO i GDZIE wpisywać, aby w pełni skorzystać z usług DOS-u. Na początek może: GDZIE.

Sprawa jest prosta. Komórki $00-$05 służą do wpisywania słów rozkazowych. W nich też dostępne są bajty statusowe informujące o wykonaniu zadania, czy też o błędzie. Następne komórki ($06-$11), to odpowiednio numery ścieżek i sektorów przekazywane jako parametry. Należy je zawsze odpowiednio ustawić PRZED wpisaniem słowa rozkazowego. Tak więc: $06 i $07 zawierać powinny numery ścieżki i sektora, które chcemy przekazać jako parametry słowa rozkazowego wpisanego do $00, $08 i $09 - słowa rozkazowego w $01, itd.

Czyżby jednak było wszystko jedno której z komórek $00-$05 używamy?

Otóż nie. Komórki te są jakby sprzęgnięte z odpowiednimi buforami w RAM. $00 odpowiada buforowi od adresu $300 (bufor #0), $01 - od adresu $400 (bufor #1), itd. Komórka $05 reprezentuje bufor, który nie jest w pamięci RAM założony. Jeżeli więc chcemy, aby sektor 18,0 znalazł się w RAM od adresu $300, to wpiszemy 18 do komórki $06, 0 do $07, $80 jako rozkaz do $00, oraz poczekamy, aż zgaśnie najstarszy bit tej komórki, aby odczytać status. Gdybyśmy chcieli wczytać sektor pod adres $700, to numer ścieżki wpisalibyśmy do $0E, numer sektora do $0F, a rozkaz do $04. Jest to chyba oczywiste. Zajmijmy się teraz drugą sprawą i powiedzmy CO można wpisywać jako słowa rozkazowe. Wiemy już, że rozkazy muszą mieć zapalony najstarszy bit (natomiast słowa statusowe - nie mogą). Ponadto najmłodszy bit oznacza, którego napędu ma dotyczyć rozkaz (czyli w naszym wypadku zawsze musi być wyzerowany). Rozkazów tych nie ma zbyt wiele. Wypiszmy najczęściej używane:

$80 - odczyt sektora

$90 - zapis sektora

$B0 - inicjowanie dysku

$C0 - przesunięcie głowicy na ścieżkę 0

$E0 - wykonanie programu w buforze

Pozostałych rozkazów praktycznie się nie używa. Tych pięć zazwyczaj wystarcza. Omówmy je może nie po kolei:

$B0 - Ogólnie można powiedzieć, że rozkaz ten służy do inicjowania dysku na żądanej ścieżce. Numer ścieżki jest też jedynym parametrem, jaki ustawić należy przed jego wywołaniem. Cóż to jednak właściwie jest to inicjowanie? W tym wypadku polega ono na odczytaniu, z jakim znacznikiem ID formatowany był dysk. Ponieważ w czasie formatowania znacznik ten zapisywany jest w nagłówku każdego sektora, wystarczy odczytać jeden taki nagłówek. Należy tutaj dodać, że po wykonaniu tego rozkazu, nawet jeżeli status brzmi O.K. nie można mieć pewności, że głowica znajduje się nad ścieżką, na której kazaliśmy inicjować dysk. Jest to spowodowane tym, że gdy stacja "nie wie" nad jaką ścieżką znajduje się głowica, nie przesuwa jej (bo i tak nie wiadomo o ile i w którą stronę), ale mimo wszystko próbuje odczytać jakiś nagłówek. Jeżeli operacja ta się uda, stacja może stwierdzić w którym miejscu dysku głowica się znajduje i wszystkie następne rozkazy będą działały poprawnie. Rozważmy może jeszcze sytuację, kiedy głowica znajduje się powyżej 35 ścieżki i stacja o tym nie wie (bo na przykład dopiero co włączyliśmy jej zasilanie). Załóżmy też, że do stacji włożyliśmy jak najbardziej poprawnie sformatowany dysk. Jak w takiej sytuacji zadziała rozkaz $B0? Nietrudno dojść do wniosku, że nie zainicjuje on dysku. Nie wolno mu przesunąć głowicy, a powyżej 35 ścieżki nie znajdzie żadnego sektora. W takim wypadku z pomocą przychodzi nam kolejny rozkaz:

$C0 - Nie przekazuje się do niego żadnych parametrów. Nie trzeba też testować błędów, gdyż nie ma możliwości aby takowe wystąpiły. Rozkaz ten przesuwa po prostu głowicę na ścieżkę zerową. (Przy obsłudze silnika krokowego przyjęło się numerować ścieżki rozpoczynając od zera. Tak więc jeżeli głowicę przesuniemy na ścieżkę zerową, znaczy to, że możemy czytać sektory ze ścieżki numer 1. Wszystko przez to, że w formacie Commodore najniższą ścieżkę nazwano pierwszą, a nie, jak we wszystkich innych formatach, zerową.) Po wykonaniu tego rozkazu stacja "wie" gdzie głowica się znajduje i od tej pory wszystkie jej przesunięcia są dozwolone.

Pokażmy może te dwa rozkazy na przykładzie:

```
*= $0400
        lda #$12
        sta $06
loop    lda #$b0
        sta $00
wait1   lda $00
        bmi wait1
        cmp #$02
        bcc ok
        lda #$c0
        sta $00
wait2   lda $00
        bmi wait2
        bpl loop
ok      rts
```

Jest to przykład inicjowania dysku. Jeżeli rozkaz $B0 zakończy się błędnie, to głowica jest przesuwana na ścieżkę 0 i wszystko zaczyna się od nowa. Program ten zakończy się, gdy dyskietka zostanie poprawnie zainicjowana.

Spróbujmy uruchomić ten program bez dysku w stacji, a po chwili włóżmy jakiś sformatowany dysk. Bez obawy, nic mu się nie stanie. Jak widać program się skończył. Warto spróbować uruchomić ten program na 1541, a później na 1571 (obojętne, czy w trybie 41, czy 71). Różnica jest wyraźna. Wynika ona z różnego sposobu przesuwania głowicy na zerową ścieżkę.

Stacja 1541 robi po prostu pewną liczbę kroków w tył, w wyniku czego po pewnym czasie głowica (a właściwie cały "wózek" do którego jest zamocowana) zaczyna uderzać w specjalny zderzak. 1571 natomiast ma wbudowaną dodatkową barierę świetlną informującą, czy głowica znajduje się na ścieżce zerowej, czy też nie. Rozkaz $C0 działa wtedy w ten sposób, że przesuwa głowicę w tył, aż do otrzymania sygnału, że bariera została przerwana, co oznacza, że w tym momencie owo przesuwanie należy zakończyć. Powyższy przykład nie pokazuje może najlepszego sposobu inicjowania dysku, pozwala jednak na lepsze poznanie rozkazów $B0 i $C0. Omówmy kolejne rozkazy:

$80 - Jeżeli dyskietka jest zainicjowana, rozkaz ten przesuwa głowicę na odpowiednią ścieżkę i próbuje czytać odpowiedni sektor. W wypadku błędu - zwraca jego numer. Jeżeli natomiast znaki ID w pamięci i w nagłówku sektora są różne (dysk nie jest zainicjowany) zwraca błąd numer $0B.

A oto przykład:

```
        *= $0400
        lda #$00
        sta $09
        lda #$01
        sta $06
loop3   jsr $f24b
        sta $08
        lda #$00
        sta $07
loop2   lda #$80
        sta $00
wait1   lda $00
        bmi wait1
        ldx #$00
        lda $09
loop1   eor $0300,x
        inx
        bne loop1
        sta $09
        lda $1c00
        eor #$08
        sta $1c00
        inc $07
        lda $07
        cmp $08
        bcc loop2
        inc $06
        lda $06
        cmp #$24
        bcc loop3
        rts
```

Program ten zlicza sumę kontrolną wszystkich sektorów dysku. Po jego zakończeniu znajduje się ona w komórce $09. W programie użyta jest procedura $F24B. Na podstawie numeru ścieżki podanego w akumulatorze oblicza ona ile na niej znajduje się sektorów. Wynik zwracany jest także w akumulatorze. Program ten jest dość prosty i nie wymaga chyba komentarza. Jedyną uwagą nasuwającą się podczas jego uruchamiania jest to, że nie jest on zbyt szybki. Za miesiąc pomyślimy jak go przyspieszyć, tymczasem zajmijmy się kolejnymi rozkazami:

$90 - Działa prawie identycznie jak $80, z tym, że zapisuje sektor na dysk zamiast go czytać. Warto dodać, że automatycznie wykonywana jest weryfikacja zapisu i nie ma potrzeby przeprowadzania jej osobno.

$E0 - Jeżeli to możliwe, rozkaz ten przesuwa głowicę na żądaną ścieżkę, po czym procesor wykonuje skok (nie JSR!!!) do pierwszego bajtu odpowiedniego bufora. Program w buforze powinien kończyć się załadowaniem akumulatora kodem zwrotnym (bit 7 musi być wyzerowany!) i skokiem: JMP $F969, lub też, jeżeli pracujemy ze stacją 1571 w trybie 71, skokiem: JMP $99B5. W wyniku tego nasz kod zwrotny umieszczony zostanie w odpowiedniej komórce ($00, $01, itd., zależnie gdzie wpisaliśmy $E0) i program główny będzie mógł kontynuować działanie.

Pokażmy to na przykładzie:

```
*= $0300
irq      jsr flash
         jsr flash
         jsr flash
         lda #$00
         jmp $f969

flash    ldy #$fe
loop1    jsr blink2
         dey
         cpy #$01
         bne loop1
loop2    jsr blink2
         iny
         cpy #$ff
         bne loop2
         rts

blink    lda $1c00
         eor #$08
         dex
         bne blink
         sta $1c00
         rts

blink2   tya
         tax
         jsr blink
         tya
         eor #$ff
         tax
         jmp blink

start    lda #$12
         sta $06
         lda #$e0
         sta $00
wait     lda $00
         bmi wait
         rts
```

Program ten uruchamiać należy skokiem do etykiety START. Etykieta IRQ musi wskazywać na pierwszy bajt bufora (np. $300, jak w tym przykładzie), gdyż tam nastąpi skok z procedury obsługi przerwania. Tego programu również nie trzeba dokładniej opisywać. Całe jego działanie polega na trzykrotnym "ładnym" błyśnięciu diodą DRIVE. Pokazuje on jednak jak używać rozkazu $E0.

Jak zwykle zachęcam do eksperymentowania. Następnym razem omówimy jeszcze kilka przykładów pokazujących do czego wykorzystać można DOS, oraz powiemy z czego składają się faktycznie takie obiekty jak znak synchronizacji, nagłówek, czy blok danych.

K.M./TABOO
Krzysztof Matula

VBS

Niedawno w reklamach na łamach różnych mniej lub bardziej profesjonalnych czasopism pojawiły się hasła pod tytułem "Super Nowość". Chodzi tu o tzw. Video Backup. Przypomniały nam one, że od ponad dwóch lat jesteśmy w posiadaniu supernowości lecz zapomnieliśmy poinformować o tym naszych Czytelników (istniejemy w papierowej wersji niecałe dwa lata). Niniejszym chcielibyśmy nadrobić zaległości.

HARDWARE

"*Video Backup System*" - tak brzmi pełna nazwa produktu niemieckiej firmy *Rossmoeller*. Urządzenie pozwala nam na zapisywanie zawartości twardego dysku lub dyskietek na taśmę magnetowidową. Do czego to się może przydać? No cóż, każdy kto choć raz utracił (najczęściej na skutek własnego, nieprzemyślanego postępowania) zawartość twardego dysku, wie jak ważne jest posiadanie w miarę aktualnej kopii jego zawartości.

Aby umożliwić wykonywanie takich kopii, mądrzy ludzie wymyślili urządzenia zwane streamer'ami. Streamer to mniej więcej coś takiego jak magnetofon (tj. Datasette) do C-64. Bierzemy sobie twardy dysk i kopiujemy jego zawartość na taśmę magnetyczną. Tyle tylko, że w porównaniu do Datasette (tego od C-64), streamer musi posiadać kilka innych cech. Do najważniejszych należy zaliczyć szybkość działania oraz pewność bezpieczeństwa zapisanych na nim danych. Ani jedna cecha ani druga nie należy do mocnych stron Datasette...

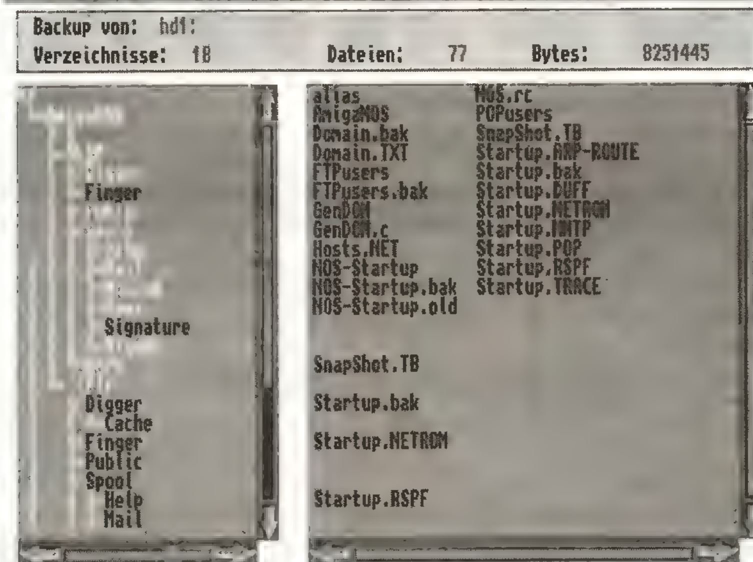
Inną dość istotną sprawą jest pojemność kasety. Dobrze byłoby mieć kasety o pojemności co najmniej równej pojemności twardego dysku, który chcemy skopiować. W przypadku Datasette, jak również innych urządzeń pracujących w oparciu o mechanizmy i taśmy magnetofonowe (pamiętamy jeszcze Spectrum'a?) występują dość istotne ograniczenia. Po prostu pasmo przenoszenia tych urządzeń nie pozwala na wystarczająco szybką transmisję danych przeznaczonych do zapisu. Potrzeba jest jednak matką wynalazków.

Wynaleziono w związku z tym specjalne urządzenia o precyzyjnych mechanizmach i zaczęto na nich zapisywać cyfrowe dane z dużą gęstością i szybkością. Specjalizowane streamery, bo o nich właśnie mowa, mają jednak kilka istotnych dla przeciętnego użytkownika wad. Po pierwsze cena, po drugie cena i po trzecie również cena. Zarówno samo urządzenie (mechanizm) jak również interface (np. SCSI) czy specjalne kasety do streamer'a są bardzo drogie. Pomysłowy Dobromir postanowił więc poszukać w domu jakiegoś innego urządzenia o wystarczająco szerokim paśmie przenoszenia. Wzrok jego padł na magnetowid ("wideło" - jak mówią niektórzy).

Gdyby tak cyfrowe dane z twardego dysku (lub dyskietki) zamienić na obraz, to możnaby taki obraz zapisać na taśmie magnetowidowej a potem należałoby już tylko rozwiązać problem transmisji w drugą stronę tj. zamiany obrazu na cyfrowe impulsy zrozumiałe dla komputera.

Tego typu rozwiązanie ma taką przewagę nad "prawdziwym" streamer'em, że bardzo często magnetowid w domu już jest, kasety do niego są stosunkowo tanie i pozostaje tylko dokupić odpowiedni interface, np. opisywany poniżej produkt firmy Rossmoeller i już możemy działać. Wspominane urządzenie skonstruowane jest w postaci kilku kabelków zakończonych wtyczkami. Jedną z tych wtyczek należy podłączyć do gniazda "serial" Amigi, drugą do magnetowidu, trzecią do wyjścia "video" (mono) komputera a czwartą na wejście "video" w monitorze. Wtyczka podłączana do gniazda "serial" zawiera w sobie całą elektronikę niezbędną do zamiany sygnałów wizyjnych na impulsy cyfrowe czyli rozwiązania w/w problemu transmisji w "drugą stronę".

Transmisja w "pierwszą stronę" czyli z komputera na





magnetowid rozwiązana jest programowo. Dołączone do zestawu oprogramowanie zamienia na życzenie dane z dysku na odpowiednie "prążki" wyświetlane na ekranie Amigi. Poprzez wyjście video Amigi obraz ten jest przesyłany na magnetowid i tam zapisywany. Program generuje dodatkowo czołówkę zawierającą informacje o tym jaki dysk (partycja) jest aktualnie zapisywany oraz datę operacji i umożliwiającą późniejszą orientację w tym co akurat zawiera dana taśma.

Zanim jednak zaczniemy zapisywać nasze dyskietki (itp) na magnetowid, możemy natknąć się na pewne trudności. Jak zwykle będzie to "problem kabelka". Otóż w Niemczech najbardziej rozpowszechnione są magnetowidy z gniazdami typu "Euro-AV" (SCART). W związku z tym wtyczka przeznaczona do podłączenia do magnetowidu, jest właśnie tego typu. Ponieważ Polska skutecznie oparła się "europeizacji" a postanowiła raczej zostać drugą Japonią, dlatego też większość magnetowidów w Polsce nie posiada gniazd "Euro-AV". Tu zaczyna się pierwszy zgrzyt. Musimy bowiem dopasować sobie kabelek do magnetowidu. Dobrze, jeżeli wiemy jak to zrobić i potrafimy posługiwać się lutownicą, gorzej gdy umiejętności te są nam obce...

Załóżmy jednak, że mamy już odpowiednie wtyczki na końcu kabelka. Teraz zaczynamy nasz pierwszy "backup". Szybko wybieramy jakąś kasetę wizyjną np. z napisem "Dynastia - odc. 2491" i stwierdzamy, że może mama nie zauważy... Wkładamy ją do magnetowidu i bezlitośnie kasujemy kolejne przygody naszych ulubionych bohaterów.

Program o nazwie VBS (Video Backup System) pozwala nam na następujące operacje:

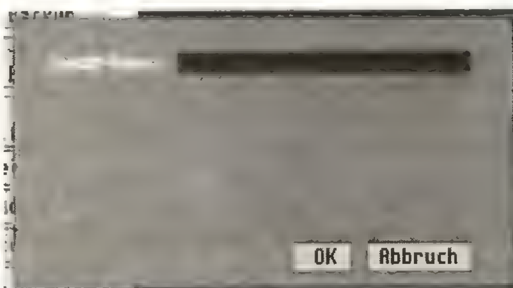
Floppy-Backup
Floppy-Restore
Dateien-Backup
Dateien-Restore
Dateien verifizieren
Bandverzeichnis...
Zeile löschen
Verz. aktualisieren
Bandverz. erstellen
Ende VBS

Floppy-Backup i Floppy-Restore pozwala nam na zapisywanie i przywracanie zawartości całych dyskietek. Następne dwie funkcje robią to samo ale z poszczególnymi zbiorami dyskietki bądź twardego dysku. "Dateien Verifizieren" to funkcja pozwalająca na weryfikację poprawności zapisu. Kolejne cztery zajmują się tworzeniem, edycją, aktualizacją i przeglądaniem tzw. spisu zawartości kasety. "Ende VBS" to niemiecki odpowiednik podstawowej funkcji niektórych programów - Quit...

Zależnie od tego jaką funkcję wybierzemy, program bądź narysuje nam strukturę "drzewka" naszego dysku i poprosi o wybór odpowiednich zbiorów, bądź zapyta się przy pomocy odpowiedniego requestera o to z którego napędu chcemy kopiować dyskietkę, ewentualnie, jeżeli wybierzemy którąś funkcję RESTORE, zapyta się o nazwę "Backup'u", który chcemy odczytać. My na pierwszy ogień posłaliśmy najmniejszą partycję twardego o rozmiarach +/- 30MB. Po zapisaniu na taśmę postanowiliśmy zweryfikować zapis.

Co prawda producent w instrukcji obsługi zapewnia, że dzięki zastosowanym skomplikowanym procedurom korekcji błędów istnieje bardzo małe prawdopodobieństwo ich wystąpienia lecz pozostawił funkcję "Verifizieren" dla tych niedowiarków, którzy chcą być całkiem pewni tego, że ich dane zostały zapisane prawidłowo. My byliśmy akurat takimi niedowiarkami... Weryfikacja wykazała kilkanaście błędów! Właściwie trudno się dziwić.

Na ekranie monitora podczas odtwarzania oprócz prążków wygenerowanych przez komputer, pojawiały się co chwilę różne "obce" spowodowane bądź zużyciem mechanicznym nośnika magnetycznego, bądź niedokładnym prowadzeniem taśmy przez mechanizm magnetowidu. O ile twarz Blake'a Carringtona pozostanie twarzą Blake'a Carringtona nawet wtedy, gdy przelecą po niej różne prążki, o tyle zakłócenie zaledwie kilku właściwych prążków przez inne spowoduje, że nasz program, obrazek czy



jakiekolwiek inne dane szybko przestaną być tym czym były. Nauczeni doświadczeniem zdobytym w kilku próbach, postanowiliśmy zmienić sprzęt. Okazało się, że dobry magnetowid i nowa, nie zużyta taśma dobrej jakości, są w stanie zadowolić VBS'a.

Zarówno nowy magnetowid VHS - Panasonic jak i stary Grundig systemu Video 2000 dały wystarczającą jakość zapisu niezbędną do poprawnego odczytania danych. Niestety nie da się tego powiedzieć o innych magnetowidach, na których próbowaliśmy wykonać operację Backup - Restore. Nawet zastosowanie nowych, markowych kaset nie dało odpowiednich rezultatów w przypadku tzw. popularnych magnetowidów. Jak zatem podsumować nasze zmagania z Video Backup Systemem?

Urządzenie to zdaje egzamin pod warunkiem posiadania dobrego (nie najtańszego) magnetowidu i zastosowania dobrych, markowych kaset. Szybkość transmisji jest zadowalająca i co jest ważne, operacje Backup czy Restore nie wymagają, podobnie jak w przypadku normalnych streamer'ów interwencji użytkownika. Pojemność jest również dosyć przyzwoita. Na czterech godzinach kasety magnetowidowej można zmieścić ok. 200MB danych. Niestety wymagania sprzętowe eliminują wielu potencjalnych użytkowników. Ktoś, kto posiada kiepskiej jakości magnetowid, może w zasadzie zapomnieć o spokojnym zapisywaniu na nim zawartości twardego dysku.

SD!

Produkt:

Video Backup System

Producent: Rossmoeller Computer Technik

Neuer Markt 21

D-5309 Meckenheim

Cena: ca. 80,- DM w Niemczech

+ duża szybkość transmisji

+ duża gęstość zapisu (pojemność kaset)

+ niska (w porównaniu do streamer'ów) cena

- wysokie wymagania sprzętowe (jakość magnetowidu)

- brak stuprocentowej pewności zapisu

Ocena KEBAB'a (1 do 6) - 3 dostateczna

FORUM

Forum jest rubryką całkowicie oddaną w Wasze ręce. Jeżeli chcielibyście podzielić się z innymi Czytelnikami Waszymi spostrzeżeniami, odkryciami, uwagami, zaprezentować króciutkie programiki, ułatwienia do gier etc. to piszcie na adres redakcji, z równoczesnym dopiskiem FORUM na kopercie. Teksty zamieszczać będziemy w formie oryginalnej. Czekamy na listy. Przyszłość tej rubryki zależy jedynie od Was.

Przesyłam Wam program - pchełkę służący do resetowania komputera. Teraz zamiast pisać: SYS 64738 wystarczy wcisnąć klawisz RESTORE.

```
199 REM *** BY SEMI OF FOI ***

200 DATA 169, 252, 141, 025, 003, 169, 226, 141

201 DATA 024, 003, 096

202 :

203 FOR I=50000 TO 50010 : READ Q : POKE J, Q : NEXT

204 SYS 50000 : PRINT "RESTORE-RESET" : NEW
```

Aby po resecie było można dalej korzystać z programu trzeba wpisać:

SYS 50000

Marcin Lewandowski - SEMI

Ułatwienia do gier nadesłane przez Jarosława Kothe

Gilligans Gold (Ocean)

POKE 17950, 0 - Bonus
POKE 21201, 173 - Życia
SYS 25532 ($63BC)

Radax (Magic Disk)

POKE 18402, 173 - Nieśmiertelność
POKE 17469, x+186 - x ilość żyć
SYS 49152 ($C000)

Master Head (Sonix)

POKE 14146, 169 - Nieskończona ilość prób
POKE 32088, x - Numer poziomu
SYS 32089 ($7059)

O Amigowym Ray-Tracingu

- słów kilka

Cóż jest tak fascynującego w rysowaniu na ekranie, przecież mając dobre, znane z przedszkola, woskowe kredki także można stworzyć dzieło sztuki. Ktoś kto zaryzykował i pokazał rodzinie swoje dokonania, usłyszał zapewne inteligentną odpowiedź, że Matejko to ty nie jesteś, ale w tym domu to na ścianie "tego" nie powiesisz. Być może duży odsetek od takich niedoszłych malarzy stanowią osoby, które postanowiły zakupić Amigę, właśnie po to, by oddać się pracy twórczej.

Zastanówmy się teraz jakich narzędzi mogą do tego celu użyć, a jest ich niemało. Pierwszy na myśl przychodzi oczywiście Deluxe Paint, jako najbardziej znany i rozpowszechniony program do tworzenia grafiki. Pojawiają się jednak pewne zastrzeżenia: primo po pierwsze trzeba być naprawdę dobrym grafikiem, żeby sklecić coś co nie spowoduje rozbicia kineskopu przez widzów i primo po drugie: malowanie przestrzennych przedmiotów powoduje masę komplikacji, a naprawdę nikłą pomocą jest możliwość zniekształceń perspektywicznych oferowanych przez ten program. Inna sprawa, że na scenie są tacy graficy, którzy wiele potrafią i ich dokonania są naprawdę rewelacyjne, przykładem może być znany wszystkim Rys z polskiej sceny, natomiast na zachodzie prym wiedzie chyba RWO z Kefrens.

To jest jednak normalne, bo ów program tak jak i dziesiątki innych pomagają tylko w jednym: tworzeniu grafiki w 2 wymiarach i to użytkownik musi zadbać o złudzenie perspektywy przy rysowaniu. To, że jakakolwiek graficzna prezentacja jest używana praktycznie wszędzie, od brudnej, wymiętej kartki sprzedawcy bananów informującej o cenie za jeden kilobajt,..., kilogram począwszy, a na czołówkach CNN International skończywszy.

Pomijając jakość, mamy do czynienia z tym samym celem: zareklamowania, ewentualnie poinformowania odbiorcy w sposób jak najbardziej przystępny i efektowny o czymś co ma właśnie nam przynieść profit, a nie konkurencji. Ponieważ jednak grafików jest zawsze za mało, pewni naprawdę inteligentni osobnicy wymyślili sposób kreacji, przy którym olbrzymią większość pracy wykonuje nasz niewolnik (zakup odbył się jednak nie na podstawie wyglądu zębów, ale na podstawie wyglądu klawiszy, których podobieństwo do zębów jest uderzające...).

Gdyby zapytać przeciętnego posiadacza komputera firmy Commodore, w jakim celu zdecydował się na wydanie pokaźnej bądź co bądź sumy, 70 procent respondentów odpowiedziałoby, że jest to najlepszy na rynku polskim sprzęt do gier, natomiast reszta prawdopodobnie jako przyczynę zakupu podałaby zamiłowanie do tworzenia komputerowej grafiki (uwaga: CBOS nie potwierdził tych danych, z braku własnych danych)

Metoda ta nazwana została swojsko "ray tracing" i stała się wydarzeniem, i choć brzmi to trochę nielogicznie, chyba nawet większym od zbudowania pierwszego komputera.

1.Koń jaki jest każdy widzi.

Samo sformułowanie ray tracing oznacza oczywiście śledzenie promieni i w pełni oddaje istotę sprawy. Zastanówmy się jednak najpierw jak to się właściwie dzieje, że widzimy to co widzimy (Ty czytelniku patrzysz właśnie w to miejsce). Proces postrzegania nie jest wprawdzie



do końca wytłumaczony, ale to i owo udało się odkryć. Na światło białe składa się promieniowanie o długości fal od 360 do 830 nanometrów. W tym dosyć wąskim zakresie znajduje się pełne spektrum barw i to właśnie światłu zawdzięczamy możliwość posługiwania się zmysłem wzroku.Bo tak naprawdę to nie widzimy przedmiotów, tylko światło, które odbijając się od ich powierzchni trafia do naszego oka (cały ten proces był omawiany na pasjonujących lekcjach biologii i fizyki, dlatego dajmy sobie z tym spokój), a otaczający nas świat jest dlatego kolorowy, że przedmioty "umieją" pochłaniać pewien zakres promieniowania widzialnego, odbijając jedynie tą część, która charakteryzuje barwę oglądanego przedmiotu. Jako przykład weźmy flagę z naszymi barwami narodowymi (przypominam: kolor biały i czerwony). Część biała odbija praktycznie cały zakres promieniowania, natomiast czerwona pochłania większość zakresu, a odbija jedynie fragment odpowiedzialny za kolor czerwony. Proste, nie ?

Wydawać by się mogło, że znając te fakty i mając w domu jakikolwiek komputer można trzymając się praw fizyki, napisać sobie program, który na podstawie danych o wyglądzie obiektów stworzy nam wizerunek nie odbiegający zbytnio od rzeczywistości. Właściwie jest to prawda, ale jest to prawda okrutna. Zauważmy bowiem, że każde źródło światła emituje we wszystkich kierunkach w ciągu sekundy potężną dawkę fotonów, z których każdy ma inną

częstotliwość wibracji co trzeba uwzględnić przy tworzeniu kolorowych obrazów.

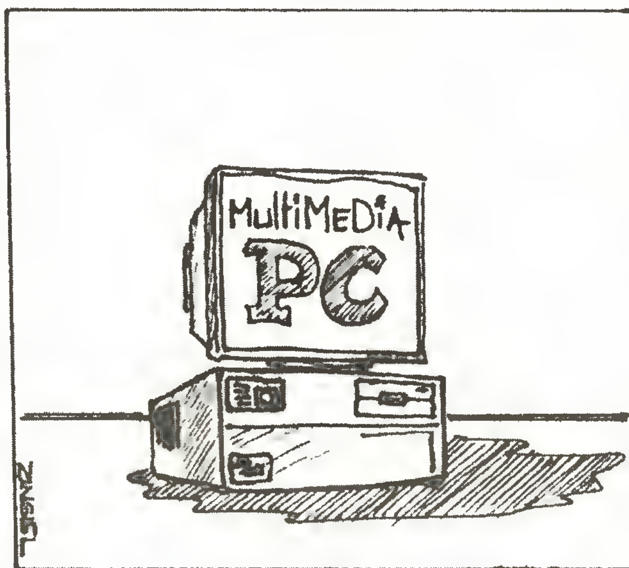
Jeśli ktoś nadal upiera się przy takim modelu, to trzeba jeszcze powiedzieć, że trzeba niestety analizować drogę WSZYSTKICH, bez wyjątku fotonów, bo niektóre z nich mają szansę wpaść do naszych oczu. Rzeczywiście, taki program nie jest trudno napisać, trzeba tylko uprzedzić potomków, żeby za 50 lat nie wyłączyły komputera dziadka, bo dziadzio chciał obejrzeć sobie oświetlony sześcian. Naszkicowana tu metoda, w odróżnieniu od innej stosowanej powszechnie, śledzi drogę promienia światła w kierunku jego rozchodzenia się. Ten właściwy ray tracing robi to dokładnie odwrotnie.

2.Śledztwo w sprawie rodziny Fotonów, czyli Kloss 'n Watson na tropie.

Trzeba najpierw stwierdzić, które fotony na pewno wpadną do naszych oczu. Obieramy sobie więc w przestrzeni płaszczyznę rzutowania o określonych wymiarach - będzie to nasz wyimaginowany ekran monitora. Dzielimy ją na małe prostokąciki, które określimy sobie mianem pixeli. Z matematycznego punktu widzenia promieniem będzie półprosta o początku w punkcie określanym mianem obserwatora, a przechodząca przez środek kolejnego pixela z płaszczyzny rzutowania.

Trzeba teraz sprawdzić przecięcia tej prostej ze wszystkimi obiektami występującymi na scenie, co wprawdzie nie jest zadaniem trudnym do zrealizowania w przypadku prostych powierzchni, staje się jednak skomplikowane i kosztowne przy powierzchniach zdefinowanych parametrycznie. Odrzucamy teraz wszystkie punkty przecięcia znajdujące się za obserwatorem, a bierzemy pod uwagę przecięcie najbliższe obserwatorowi. Jeżeli promień napotkał obiekt, który nie daje odbić zwierciadlanych i jest nieprzezroczysty to wystarczy jedynie wyznaczyć kolor w tym miejscu i kontynuować proces. W pierwszych programach realizujących ray tracing tak właśnie postępowano, co dawało mizerne niestety efekty.

Przy niskiej rozdzielczości bardzo wyraźnie widać pikselową strukturę obrazu, zauważalną zwłaszcza w miejscach ostrych przejść kolorów. Mamy wtedy dwa możliwe wyjścia z sytuacji: albo zwiększamy rozdzielczość obrazu uzyskując gładkie krawędzie i długi plik z obrazem, albo stosujemy "anti-aliasing", czyli w wolnym przekładzie wygładzanie krawędzi. Jedną z metod jest prowadzenie promienia nie tylko przez środek piksela, ale również przez jego rogi i uśrednienie kolorów otrzymanych z każdego promienia. Ów sposób postępowania jest bardzo prosty w realizacji i w większości przypadków wystarczający.



— BEZ SŁÓW —

Czy wiedzieliście, że...

Firma IBM (tak, ta sama, która stworzyła niegdyś pierwowzór wszystkich jedynie słusznych komputerów) proponuje swoim klientom zestawy typu Multimedia oparte o komputery IBM? O tym jak doskonałe są te zestawy mogli przekonać się zwiedzający wystawę targową IBM'a na ostatnich targach CEBIT '93. W Hali 2 odbywały się m.in. multimedialne prezentacje dotyczące IBM'owskiego systemu operacyjnego OS/2. Prezentacje bardzo się zwiedzającym podobały ale tylko niektórzy wścibscy zauważyli, że IBM zdecydował się wykorzystać, do tego celu zabawkowy komputer Amiga oraz sterujący wszystkim program "Scala Multimedia"...

3.Przesłuchanie, czyli lampa prosto w oczy.

Warto zastanowić się teraz w jaki sposób określić natężenie oświetlenia w punkcie znalezionego przecięcia. Zakładamy na początku, że mamy do czynienia z punktowymi źródłami światła, czyli takimi, które emitują światło we wszystkich kierunkach. Powierzchnia, której oświetlenie badamy charakteryzować się będzie (na razie) jedynie wektorem prostopadłym do niej. Wektor taki zwany jest także wektorem normalnym.

Gdy wiemy już to wszystko, okazuje się, że natężenie światła zmienia się tak jak cosinus kąta między wektorem normalnym, a wektorem padania światła. Ktoś domyślny stwierdzi pewnie, że w rzeczywistym świecie nie ma zbyt wielu przedmiotów, które mają idealną powierzchnię, zawsze są jakieś pofałdowania, czy zniekształcenia. Wspaniale ! Właśnie oto chodzi. W bardzo prosty sposób można nieznacznie zakłócać wektor normalny, tak, aby otrzymać bardzo realistycznie wyglądające, chropowate przedmioty. Nazwa tej techniki brzmi "bump mapping".

Owo rozłożenie wypukłości można uzyskać stosując generator liczb losowych, ale wygodniej jest posłużyć się wczytanym z dysku niewielkim obrazem zawierającym rozłożone w miarę regularny sposób pixele. Zależnie od możliwości oferowanych

przez program, każdy taki pixel może być przedstawiony jako mała kulka, bądź też jako walec, czy spirala.

4.Dobre przebranie gwarancją dobrego w skutkach śledztwa.

Właściwie główny nacisk w programach do ray tracingu położono na maksymalne zbliżenie się do rzeczywistości. Jednak w niektórych przypadkach jest to bardzo trudne i nieopłacalne. Weźmy na przykład marmur czy korę drzewa. Zamiast ślęczeć wymyślając algorytmy, które przedstawią nam taką powierzchnię, prościej jest posłużyć się gotowym, zeskanowanym obrazem i w odpowiedni sposób "nałożyć" go na rysowany przedmiot. Proste, a jakie efektowne.

Metoda ta nazwana została teksturowaniem (ang. texture, czyli wzór). Trzeba tylko pamiętać, że nawet jeden taki obrazek zajmuje sporo miejsca w pamięci, co uniemożliwia sprawne wykorzystanie tej techniki na beznadzienie wolnych komputerach z mikroskopijną ilością pamięci (6 mb z A4000 to przy ray tracingu prawie nic...). Czy można coś na to poradzić ?

A jakże ! Trzeba tylko poświęcić jedno, by zyskać na drugim. Można bowiem zrezygnować z wysokiej wierności odtwarzanej powierzchni, by uzyskać sporo wolnej pamięci. Popatrzmy na przykład na drewniane słoje, czy szachownicę. Algorytm opisujący kolor takiego przedmiotu jest bardzo prosty i nie wpływa zbytnio na ogólny czas tworzenia obrazu (a w każdym razie, jest na pewno równy czasowi nałożenia gotowego obrazu). Pojawiają się więc tekstury matematyczne, które są funkcjami dwóch lub trzech zmiennych. Główną zaletą są zmienne parametry funkcji co daje nam możliwość łatwego eksperymentowania z modelowanym przedmiotem, a co w przypadku zwykłego nakładania obrazu jest praktycznie niemożliwe. Poza tym, plik z taką matematyczną teksturą jest parę razy krótszy od odpowiadającemu mu gotowemu obrazowi. Teksturę można i trzeba przekształcać razem z animowanym obiektem, czyli przesuwać,obracać, ewentualnie skalować.

Poza tym nakładając teksturę trzech zmiennych na obiekt złożony z kilku prostych brył, nie ma problemu z dopasowaniem brzegów tekstury (na przykład słoi w drewnie) do siebie.

5. Dobry detektyw powinien mieć kamizelkę fotonoodporną i umiejętność szybkiego ukrywania się w cieniu.

Ktoś uważny mógłby teraz zapytać: gdzie w tym wszystkim było właściwie jakiekolwiek śledzenie promieni ? Przecież zawsze był tylko jeden promień, który albo trafiał w przedmiot, albo go omijał. A co się dzieje w przypadku, gdy chcemy narysować na przykład jakiś metalowy przedmiot, w którym odbija się całe otoczenie. Przyda nam się tutaj znajomość prostego prawa fizycznego, które stwierdza, że kąt odbicia promienia od powierzchni równy jest kątowi padania liczonym między wektorem padania promienia i wektora normalnego do powierzchni.

Gdy już obliczymy kierunek tego nowego promienia, możemy znów śledzić jego drogę, aż do napotkania kolejnego przedmiotu, czyli kolejnego odbicia itd. Co jednak zrobić gdy promień będzie odbijał się w niekończoność (wpadnie dajmy na to do półprzezroczystego sześcianu). Trzeba wtedy brutalnie przerwać jego drogę po przekroczeniu pewnej, z góry ustalonej liczby odbić. Z reguły liczba ta jest ustawialna we wszystkich programach do ray tracingu, gdyż zwiększając ją tracimy niestety na szybkości tworzenia obrazu. Zwykłe zwierciadlane odbicia to jeszcze nie wszystko co oferują nam typowe programy "śledzące" (wbrew pozorom nie są używane przez Urząd Ochrony Państwa).

Kolejną prostą do zrobienia rzeczą są cienie. Postępowanie jest następujące: gdy już znajdziemy przecięcie promienia z obiektem, musimy sprawdzić czy do tego miejsca w ogóle dochodzi światło, czyli w linii prostej do źródła światła nie powinien znajdować się żaden obiekt, jeśli jednak znalazł się takowy to badane miejsce przecięcia jest nie oświetlone (w rzeczywistości nie wygląda to tak różowo, światło może się przecież odbijać od różnych przedmiotów i w ten sposób oświetlać inne obiekty, ale o tym za moment).

Cóż jeszcze można zrobić ? Przecież nie wszystko jest nieprzezroczyste, prawda ? Jest to w sumie proste do zrobienia, trzeba tylko rozbić śledzony promień w miejscu znalezionego przecięcia na dwa: odbity i przechodzący. Ten drugi nie jest już równoległy do pierwszego, jego zakrzywienie zależeć będzie od stosunku gęstości dwóch ośrodków: otoczenia i materiału z jakiego wykonany jest obiekt (np. szklana kula w wodzie).

Wygląda to bardzo efektownie, a jedyny problem jaki pojawia się przed programistą to sensowne zaprojektowanie struktury "drzewa" śledzonego promienia. Jest to niezbędne, gdyż oba promienie mogą znowu się odbijać, przenikać itp. W każdym punkcie przecięcia trzeba obliczyć natężenie światła, by w końcu posumować wszystkie natężenia i "wymieszać" je w odpowiednim stosunku.

6.Gdy umiesz już ostro zgłębiać zagadki kryminalne, zgłoś się na najbliższy posterunek policji.

Omówione tu metody generowania realistycznych obrazów nie są już w dzisiejszych czasach czymś rewelacyjnym. Są to po prostu standardowe opcje oferowane przez popularne wśród użytkowników Amigi ray-tracery. Powiedzmy teraz kilka słów o tym czego brakuje tym najprostszym programom (zaliczam tutaj i Imagine i Real3d v.1.4, a dlaczego... o tym za chwilę). Ważną rzeczą dostępną w profesjonalnych programach dla stacji graficznych (firm Sun i Silicon Graphics) jest głębia ostrości.

Porównując bowiem dwa obrazy, różniące się tylko tym, że jeden wygenerowano bez głębi

ostrości, można powiedzieć, że jest to rzecz zbliżająca nas niebiezpiecznie blisko do granicy rozróżnialności zwykłego świata od tego "cyfrowego". Owa niesamowita głębia ostrości polega na symulowaniu soczewki przez którą przechodzą początkowe promienie, zaginają się i dzięki temu obszary znajdujące się daleko od punktu na który patrzy obserwator, są nieostre.

Zastanówmy się teraz dzieje się w przypadku fotografowania szybko poruszających się obiektów. W zależności od czasu naświetlania ujrzymy mocniej lub słabiej rozmyty obiekt i tylko dzięki temu możemy rozpoznać, że był on w ruchu, mimo tego, że oglądaliśmy tylko jedną klatkę z jego trajektorii. Dokładnie ten sam efekt można uzyskać przy generowaniu animacji, co zostało już umieszczone w bardziej poważnych programach.

Efekt ten określany jest mianem "motion blur" co można tłumaczyć swobodnie jako rozmycie w czasie ruchu. Jest jeszcze jeden ważny powód dla którego stosowany jest motion blur. Wyobraźmy sobie, że przygotowujemy animację szybko lecącego myśliwca, który jest obserwowany przez nas w dosyć szerokim polu widzenia. Dopóki znajduje się daleko od obserwatora, wszystko jest w porządku, ale wystarczy, że będzie tak blisko, że na jednej ramce animacji jeszcze będzie widziany jako bardzo duży obiekt, a na następnej już znajdzie się za obserwatorem, czyli zniknie z jego pola widzenia. Inny przykład. Tworzymy tym razem animację jadącego samochodu. Przydałoby się żeby kręciły mu się koła, prawda ? Nic prostszego.

Oglądając jednak już gotową scenkę, zauważamy, że koła kręcą się do tyłu ! W takich i innych przypadkach stosuje się motion blur, który jest swego rodzaju anti-aliasingiem. Zwykle tym terminem określa się wygładzanie krawędzi, w przypadku ruchu, piksele zostają zastąpione przez ramki animacji, natomiast stopień rozmycia zależy od prędkości poruszającego się obiektu.

7.Dobry śledczy udaje się najpierw do skomputeryzowanego archiwum (tylko tam podają alkoholowe drinki).

Weźmy teraz na tapetę najpopularniejsze programy realizujące ideę ray tracingu. Zaznaczę tylko na wstępie, że będą to moje osobiste odczucia związane z użytkowaniem owych programów, co oznacza, że nie każdy musi się z nimi zgodzić. Ale to już tak jest. Każdy ma swój ulubione film, piwo, typ dziewczyny, a w przypadku Amigowców dochodzi jeszcze ulubiony raytracer. So (pol. więc) w szranki stają : Real 3d v.1.4 i Imagine v.2.0. Programy które powinny robić mniej więcej to samo, nie zawsze muszą tak samo wyglądać. Zasada ta spełnia się w tym przypadku w 256 procentach. Zacznijmy od edytora. W Realu mamy właściwie jeden główny segment programu, czyli edytor główny, w którym to możemy modelować obiekty, materiały, tworzyć animacje, no i oczywiście oglądać scenę w trzech różnych rzutach. By zobaczyć ją z punktu widzenia obserwatora, można szybko i bezboleśnie przenieść się na drugi ekran, gdzie za pomocą suwaków można "kręcić obserwatorem". Oznacza to jedno.

Operacja ta dokonywana jest w czasie rzeczywistym ! Oczywiście obiekty przedstawione są za pomocą linii prostych, ale dzieje się to zadziwiająco szybko. Co by nie mówić, funkcje w edytorze Reala są wykonywane naprawdę szybko i pewnie. Dla przeciwwagi powiedzmy coś teraz o Imaginie. Całość podzielona jest na kilka edytorów: Detail służący do modelowania obiektów, nadawania im atrybutów, itp, Stage z kolei ma za zadanie wspomóc rozmieszczenie obiektów na scenie oraz Anim, który w naprawdę wygodny sposób pozwala tworzyć animację. Edytory Imagina mają jedną, zasadniczą wadę, są skandalicznie wolne i na zwykłej A500 można było się zgubić, bo nie wiadomo było, co właściwie ten Imagine w danej chwili robi. A może już się powiesił ?

Poza tym wszystko jest OK, powiem więcej, scena nad którą właśnie pracujemy jest przedstawiona znacznie przejrzyściej niż w Realu, pomaga w tym także jednoczesność czterech rzutów (trzy płaskie i jeden perspektywiczny z usuniętymi niewidocznymi liniami). Największe jednak różnice znajdują się w samym module służącym do generowania obrazów.

Otóż Imagine zna tylko dwa rodzaje powierzchni: trójkąty i kule, które są właściwie niewielkim dodatkiem, a to z tego względu, że jedyne możliwe przekształcenie kuli to skalowanie ! Zapewne chodziło o to, że wszyscy którzy chcą szybko przetestować program (na przykład na targach) stawiają kilka kulek, jedną lampkę i stwierdzają, że wszystko działa. Poprzednia wersja Imagine nie miała tak "dużych" możliwości i oferowała kulki złożone z trókątów. Jak to wyglądało, nie trzeba chyba opowiadać. Im większa była taka sfera, tym łatwiej zauważało się ostre krawędzie. Antidotum było proste. Można przecież użyć większej ilości mniejszych trójkątów. Właściwie twórcy powinni umożliwić użytkownikowi stworzenie sfery z nieskończoną ilością trójkątów o nieskończenie małych wymiarach, co powoduje rozciągnięcie czasu pracy komputera do nieskończoności, co z kolei sprawia, że kupujemy wersję Imagine 2.0.

Ponarzekałem trochę na te trójkąty, które mają jednak parę zalet. Po pierwsze łatwo się znajduje przecięcia z promieniami, a po drugie przy obiektach o niewielkich rozmiarach trudno zauważyć ich pofragmentowanie. Co oferuje nam jeszcze Imagine ? Ma przede wszystkim tekstury, których wybór już w czasach wersji 1.0 był bardzo szeroki: drewno, fale, szachownica, maskowanie wojskowe, różnego rodzaju plamy, punkty czy też przejścia od jednego koloru do drugiego. W dodatku niedawno dotarł do Polski pakiet zmiennoprzecinkowych tekstur o nazwie Essence, których ilość, a dokładnie 65, po prostu przytłacza. Wybór jest naprawdę sze-

roki, a ciekawostką jest możliwość teksturowania fraktali.

Było o Imagine, teraz parę słów o Real'u. Tu mamy do czynienia z sytuacją dokładnie przeciwną. Dosyć duży wybór różnego rodzaju brył: elipsoidy, stożki, walce, prostopadłościany, do tego możliwość modelowania obiektu za pomocą krzywych, a wszystko po to by uzyskać gładkie powierzchnie bez uciekania się do aproksymacji trójkątami. To co trochę załamuje w tym programie, to brak arytmetycznych tekstur i nieciekawy sposób nakładania obrazu na obiekty. Dobrze natomiast zrobiony jest bump mapping, który wygląda bardzo realistycznie. Kolejny plus dla Real'a to właśnie wysoka jakość tworzonych obrazów, także dzięki prawdziwemu anti-aliasingowi. W Imaginie jest on również dostępny, ale w postaci bardzo uproszczonej, przypominającej opcję Smooth z Deluxe Painta.

To co według mnie przemawia na korzyść Imagina to bardzo dobry edytor do animacji, dzięki któremu nie ma problemu ze skomplikowanymi nawet pomysłami, typu obserwator leci wzdłuż jakiegoś krzywoliniowego toru, patrząc cały czas na poruszający się w innym kierunku obiekt. Mamy do dyspozycji także kilka efektów animacyjnych, które pozwalają na eksplozję obiektu (znana kulka ze znienawidzonej czołówki Reklamą), toczenie, falowanie, itp. Świetny jest też pomysł dotyczący ruchu stawów czy zawiasów. Dzięki Cycle Editor prostym zadaniem jest zrobienie obracającej się ręki, czy latającej podpaski machającej wesoło skrzydełkami.

8.A co w przyszłości ?

No właśnie. Ukazał się wreszcie długo zapowiadany Real3d 2.0 i ma olbrzymią szansę posłać do lamusa opisane wyżej programy. Zawiera bowiem prawie wszystko to co zostało do tej pory wymyślone w dziedzinie ray tracingu. Jest po prostu genialny. Ma tylko jedną wadę, dusi się na standardowej Amidze 4000 z jej 6MB RAM'u...

Robin/WFMH

P.S. Dla maniaków którzy chcieliby jednak uruchomić ów "nieskomplikowany" programik na A500 bez rozszerzenia pamięci, mała informacja. Główny blok programu ma blisko 1 MB długości,wymaga Kickstartu przynajmniej 2.0, a producent oferuje ów program tylko w wersji zmiennoprzecinkowej.

Assembler na C-64

SOFTWARE

Czy kiedyś zastanawialiście się jak to się dzieje, że w niektórych programach obraz pojawia się także poza widocznym obszarem ekranu czyli na tzw. borderze? Ojej! Ktoś powie... Znowu będą podawali procedury na otwieranie borderu...

Niezupełnie! Otóż faktycznie gotowe procedury na "otwarcie" border'ów można znaleźć w wielu czasopismach. Niewiele jednak można znaleźć na temat tego DLACZEGO właśnie taka procedura spowoduje zniknięcie ramki. My chcielibyśmy dzisiaj zająć się właśnie tą bardziej teoretyczną stroną zagadnienia. Wiemy wszyscy zapewne już (powtarzaliśmy to wielokrotnie) jak tworzony jest obraz na ekranie monitora i jak generowany on jest przez komputer.

Mam tu na myśli fakt, że składa się on z pixeli, które tworzą linie rastrowe a te z kolei są w odpowiedniej kolejności "wykreślane" na ekranie tworzą nam cały obraz (raster). Wiemy również z poprzednich lekcji, że pod adresem $D012 znajduje się rejestr VIC'a o nazwie RASTER, w którym znajduje się zawsze wartość odpowiadająca numerowi aktualnie generowanej linii rastrowej. Wiemy również, że istnieją dwa rejestry o nazwach SCROLX i SCROLY pod adresami $D016 i $D011 odpowiednio. Wiemy także (a jeżeli nie wiemy to za chwilę się dowiemy), że bit numer trzy w rejestrze SCROLY odpowiada za to ile wierszy znaków będzie widoczne na ekranie.

W momencie gdy bit ten jest skasowany (0) to na ekranie będzie widoczne tylko 24 wiersze znaków. Normalnie (tzn. po resecie) bit ten jest ten ustawiony i widoczna część ekranu zawiera 25 wierszy. Podobną funkcję posiada bit numer trzy z rejestru SCROLX. Skasowanie tego bitu (po resecie

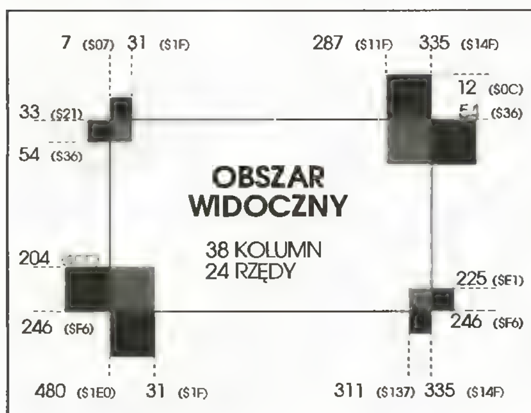


jest ustawiony) pozwala na zawężenie widocznej części ekranu do 38 kolumn. Normalnie na ekranie widoczne jest 40 kolumn, jednak ze względu na możliwość wykonywania płynnego scrollowania przy użyciu tego rejestru, konstruktorzy VIC'a umieścili weń możliwość sprzętowego zawężenia widocznej części ekranu do 38 kolumn tak aby scrollowane znaki mogły pojawiać się i znikać jak gdyby za zasłoną. Podobny cel przyświecał konstruktorom przy instalowaniu funkcji zawężania ekranu w pionie.

Przydaje się ona podczas scrollowania zawartości ekranu w kierunku góra-dół. Zresztą temat ten jest nam dobrze już znany. W końcu kilka numerów temu pisaliśmy naszego pierwszego scrolla. Nikt z konstruktorów chyba jednak nie przypuszczał, że znajdą się tacy koderzy (z grupy 1001 crew), którzy wykorzystają te bity do zupełnie innych celów. Co zatem przyszło do głowy programistom?

Otóż znając zasadę generowania obrazu przez VIC'a, wymyślili oni coś takiego: W momencie, gdy VIC kończy generować obszar widoczny, przechodzi w tzw. Vertical Blank Mode czyli mówiąc bardziej po polsku: włącza border. Spójrzmy teraz na rysunki umieszczone poniżej. Widać tam, że obszar widoczny kończy się na linii rastrowej nr $FA w przypadku trybu 25 wierszy oraz na linii numer $F6 w trybie 24 wierszy.

Co się w takim momencie dzieje? Otóż VIC po wygenerowaniu ostatniej linii obszaru widocznego musi "włączyć" border.



Ale jeżeli ekran jest zawężony do 24 wierszy, to musi to zrobić o cztery linie rastrowe wcześniej niż w przypadku normalnego (25-cio wierszowego) ekranu.

Wyobraźmy sobie teraz taką sytuację: VIC wygenerował linię $F6 i nie włączył borderu gdyż znajduje się aktualnie w trybie 25 wierszy i ma jeszcze 4 linie czasu. Teraz, dajmy na to w linii o numerze $F8, przełączamy VIC'a w tryb 24 wierszy. I co się dzieje? Nic! VIC wygeneruje linię $FA i... nie włączy borderu gdyż w trybie 24 wierszy, border włącza się o 4 linie wcześniej... Efekt jest taki, że dzięki temu prostemu zabiegowi programowemu VIC "zapomina" o tym, że powinien był włączyć border!

Oczywiście po tym jak zostanie już wygenerowana linia $FA i układ zabierze się za następne, musimy gdzieś jeszcze przełączyć VIC'a ponownie na tryb 25-cio wierszowy, tak aby podczas tworzenia następnego obrazu można było ponownie go oszukać. Co następnie możemy zrobić? Niestety nie możemy wstawić tam żadnych znaków trybu tekstowego ani obrazka w którymś z trybów graficznych.

Możemy natomiast umieścić tam... Sprite'y. Widać doskonale na rysunkach, jakie wartości współrzędnych należy wykorzystać aby umieścić sprite'y w żądanym obszarze borderu. Jest jeszcze jeden drobiazg. Otóż istnieje pewna właściwość VIC'a, która powoduje, że po "otwarciu" borderu obszar ten jest wypełniony sekwencją pixeli odpowiadającą kombinacji bitów ostatniego bajtu w danym banku graficznym VIC'a. Co to znaczy? Otóż przestrzeń adresowa C-64 to 64 kilobajty. VIC natomiast jest w stanie "widzieć" na raz tylko 16 kilobajtów. W związku z tym cała pamięć została, z punktu widzenia VIC'a podzielona na cztery tzw. banki. Po włączeniu zasilania i każdym resecie, VIC ma dostęp do najniższego (zerowego) banku pamięci. Bank ten obejmuje 16 kilobajtów od adresu $0000 do $3FFF włącznie. Bank pierwszy to obszar od $4000 do $7FFF, drugi obejmuje pamięć od $8000 do $BFFF i ostatni (trzeci) sięga od $C000 do $FFFF. Przedstawia to również poniższa tabelka. Skąd jednak wiemy jaki bank pamięci obsługuje aktualnie VIC i jak w/w banki możemy przełączać? Otóż okazuje się, że pod adresem $DD00 znajduje się rejestr o nazwie CI2PRA. To skrót od "CIA 2, PORT A" Jest to rejestr układu CIA2, gdzie dwa najniższe bity (0 i 1) umożliwiają nam wybranie banku graficznego VIC'a.

BANK	Obszar	CI2PRA ($DD00)
0	$0000-$3FFF	XXXXXX11
1	$4000-$7FFF	XXXXXX10
2	$8000-$BFFF	XXXXXX01
3	$C000-$FFFF	XXXXXX00

Po tej małej wycieczce w krainę banków VIC'a, wracamy do naszych borderów. Otóż w banku zerowym ostatni bajt znajduje się pod adresem $3FFF. Zależnie od tego jaka jest wartość tego bajtu (kombinacja bitów), taki wzorek otrzymamy na ekranie w obszarze borderu. Najczęściej pożądanym dla nas efektem będzie brak jakiegokolwiek wzorka. W związku z tym należy do ostatniego bajtu aktualnego banku wpisać wartość $00. Mając te wszystkie informacje, możemy już przystąpić do otwierania borderów.

Napiszmy zatem na zakończenie krótką procedurkę.

```
SEI
LDA #$7F
STA $DC0D
STA $DD0D
LDA $DC0D
LDA $DD0D
LDA #$01
STA $D01A
STA $D019
```

Jeżeli uważnie czytaliśmy

poprzednie odcinki to nie trzeba tłumaczyć co robi te kilka linijek. Teraz ustawimy przerwanie rastrowe na linię $F8, czyli dokładnie pomiędzy $F6 i $FA.

```
LDA #$F8
STA $D012
LDA $D011
AND #$7F
STA $D011
```

Dobrze! żebyśmy jeszcze widzieli, że faktycznie zniknie nam border, ustawimy sobie różne kolory tła i ramki. Zrobimy sobie na przykład żółty border...

```
LDA #$07
STA $D020
```

... i czarne tło

```
LDA #$00
STA $D021
```

Ponieważ mamy już w akumulatorze wartość $00, to od razu wepchniemy ją do ostatniego bajtu w aktualnym banku VIC'a. W naszym przypadku będzie to (zakładam, że dopiero co włączyliśmy komputer) bank zerowy czyli adres $3FFF.

```
STA $3FFF
```

Teraz musimy jeszcze powiedzieć systemowi gdzie ma szukać naszej procedury obsługi przerwań.

```
LDA #<NOWEIRQ
STA $0314
LDA #>NOWEIRQ
STA $0315
```

Uwaga! Startujemy...

```
CLI
```

No i powracamy da BASIC'a jak gdyby nigdy nic.

```
RTS
```

Zanim jednak uruchomimy naszą procedurkę, musimy jeszcze napisać jej drugą część, tą która będzie wykonywana w czasie przerwania.

```
:NOWEIRQ
LDA $D019
STA $D019
```

Powyższe dwie linijki potwierdzają VIC'owi obsługę przerwania (ale to już wszyscy wiemy). Teraz musimy szybko przełączyć go w tryb 24 wierszy.

```
LDA $D011
AND #$F7
STA $D011
```

Mamy to z głowy! Pozostaje tylko odczekać aż zostanie wygenerowanych kilka kolejnych linii rastrowych. Najprościej zrobimy to w ten sposób, że odczekamy aż rejestr RASTER przeskoczy na zero...

```
:PETLA
LDX $D012
BNE PETLA
```

Teraz, ponieważ ciągle jeszcze mamy w akumulatorze wartość pobraną z rejestru SCROLY ($D011), więc ustawimy w niej trzeci bit

```
ORA #$08
```

i wstawimy tą wartość z powrotem do rejestru.

```
STA $D011
```

Operacja ta przełączy nam ponownie tryb pracy na 25 wierszy aby w następnej ramce można było całą czynność powtórzyć. Na koniec jeszcze skok do standardowej obsługi przerwań.

```
JMP $EA31
```

I już po wszystkim. Następnym razem zajmiemy się bocznymi borderami i umieścimy sobie sprite'y.

SD!





Giddy

Czy pamiętacie grę zatytułowaną "Dizzy"? Zapewne tak, bowiem przygody wesołego jajka były kontynuowane w trzech kolejnych grach. Giddy nie jest bynajmniej kontynuacją poprzednich programów, ale już na pierwszy rzut oka widać, iż autor bardzo je lubił i czerpał z nich natchnienie. Główny bohater, dziwny stworek z przerośniętymi rączkami wędruje wszerz i wzdłuż zmagając się przy tym z przygotowanymi przez autora zasadzkami. Czasami rozwiązanie takiej zagadki to kwestia chwili, niekiedy zaś trzeba mocniej pogłówkować (np: na co przyda się atari st znalezione na śmietniku) w czym przydaje się wprawa wyniesiona właśnie z pierwowzoru. Myślę, że gra ta znajdzie wielu zwolenników, którzy zechcą razem z Giddym zmagać się z różnymi przeciwnościami. Co ciekawsze jest to gra typu Public Domain zatem pograć w nią (lub przegrać koledze) można z czystym sumieniem.



Universal Warrior

Miłośników gier labiryntowych, w których to głównym celem jest dotarcie do określonego punku

w wyimaginowanym labiryncie (przybierającym często postać rozmaitych miast, lasów tudzież innych, bliżej nieokreślonych form) jest zapewne sporo. Dla nich to właśnie powstała kolejna gra z tego cyklu zatytułowana Uniwersal Warrior. Owym wojownikiem jest



niewielki robocik, który przedziera się przez poszczególne etapy tej gry, zmagając się przy tym z wrogimi mu "tubylcami", zaś jedynym słusznym argumentem w bliższej z nimi konfrontacji jest oczywiście działko w który nasz bohater jest uzbrojony.

Ewentualne uszkodzenia robotka pokazują specjalne wskaźniki. zaś niektóre z nich są oczywiste i bez wskaźników (np: uszkodzenie "kamer" powoduje, że dalsza gra staje się znacznie utrudniona przez zakłócenia wizji i jedynie wizyta w warsztacie pozwala na dalsze potyczki). Aby podreperować nadwątlone "zdrowie" po przejściu



każdego etapu możemy dokonać ewentualnych napraw, wymienić lub sprzedać zbędne oprzyrządowanie i podzespoły lub przy odrobinie szczęścia wygrać parę groszy w "jednorękiego". W sumie nic odkrywczego, ale pograć można.

Nicky Boom II

Tym razem miałem wątpliwą przyjemność zapoznać się bliżej z grą Nicky Boom II. Główny bohater, Nicky prawdopodobnie, porusza się po krainie pełnej większych od niego much, ruchomych muchomorków i pluszowych miśków, zbierając przy tym rozmaite cukierki, jabłuszka i inne rzeczy tego typu, których w takich głupawych grach jest pełno. Irytująca jest niedbałość autorów o takie podstawowe rzeczy w grze jak np. animacja (którą rozwiązaną za pomocą kilku faz ruchu danej postaci). Przy tym idiotyczny (żeby nie powiedzieć debilny) uśmiech głównego bohatera wyzwala u grającego agresję już po pierwszych kilku sekundach gry. Szczerze odradzam.

Lost Vikings

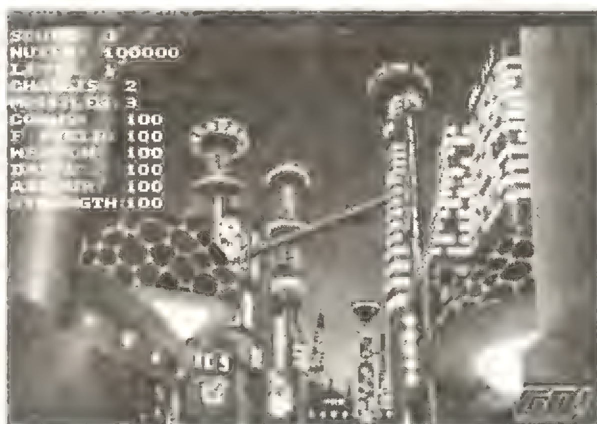
Na zakończenie gra o przygodach trzech Vikingów w... bazie kosmicznej gdzie zostali przetransportowani przez kosmitów z samego serca ich rodzinnej wioski. Sukces w ich poczynaniach może zapewnić jedynie kolektywne współdziałanie na polu walki. Każdy z nich posiada jakąś szczególną umiejętność, która w połączeniu z uzdolnieniami kolegów da zamierzony rezultat. Ponieważ gra jest przeznaczona dla jednej osoby przeto w danym momencie możemy "działać" tylko jednym Vikingiem co wymusza planowanie dalszych akcji z uwzględnieniem pozostałych bohaterów.

Scrabble

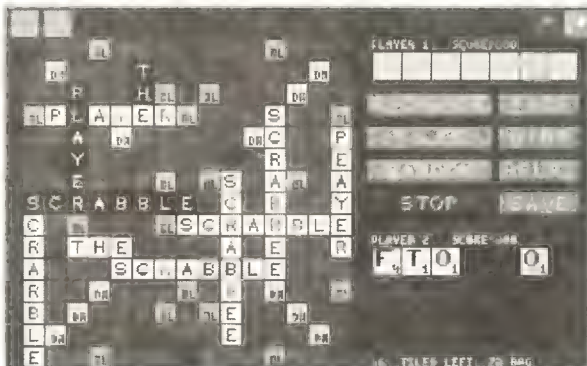
Yet another implemenation of well-known

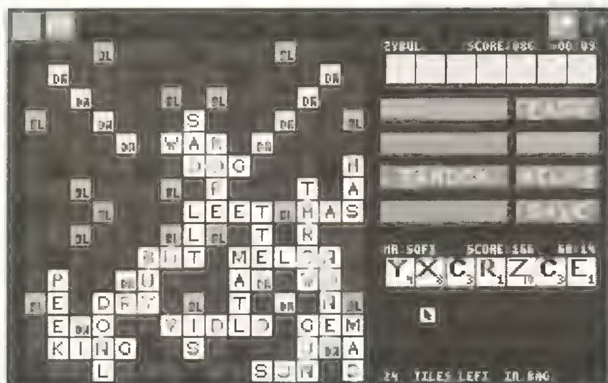


"Scrabble" has arrived to Kebab's offices. The fans of this sort of games will know what the thing is about as rules of the game remained unchanged. To put things simply, you have to fill crossword with words which are built from randomly appearing letters. When you put the letter in correct place some points are added to your score, and that's about all. In



comparision to earlier versions this one surprises us with huge help and nice game design. Loads of options let you configure the game as you wish and makes it very playable, indeed. Some English knowledge is required to make "Scrabble" interesting for you, and that's why we decided to write this small description in this so popular language.





"Conflict Europe"



Conflict Europe jest typową grą strategiczną, symulującą współczesny, zbrojny konflikt europejski. Gra ta została wydana przez firmę Mirrorsoft. Walka toczy się przy udziale broni konwencjonalnej i atomowej. Obie walczące strony, czyli państwa układu warszawskiego i NATO, decydują się na zbrojne starcie. W tym momencie odpowiedzialność za losy wojny spada na Twoje barki. Wcielasz się w głównodowodzącego jednej ze stron. Od Ciebie będzie zależało to, czy wojna zakończy się zwycięskim marszem wojsk radzieckich na zachód, czy też wojska układu NATO nie pozwolą na to i zatrzymają Armię Czerwoną wraz z sojusznikami.

Może również zaistnieć taka możliwość, że nikt z graczy nie zostanie zwycięzcą. Odpalenie całego arsenału broni atomowej jednej z walczących stron i kontra systemu szybkiego reagowania drugiej strony, może doprowadzić do ogólnoeuropejskiej katastrofy ekologicznej, zimy popromiennej, a co się z tym wiąże to każdy chyba wie...

(krótko mówiąc, jest to całkowita degradacją życia na naszym kontynencie). Gra rozpoczyna się od wyboru walczącej strony. Kliknięcie prawym przyciskiem myszy powoduje zmianę symbolu na okładce książki dowódcy. Oczywiście młot i sierp to wojska układu warszawskiego. Następnie do wyboru mamy kilka scenariuszy gry (Opening Gambit, Out of Step, After the INF, Starwars Option, What If?). W zależności od wybranego scenariusza będą występowały trzy zasadnicze różnice; rozkład wojsk i ich liczebność w momencie rozpoczęcia konfliktu, sojusznicy (głównie chodzi o sojuszników Armii Czerwonej czyli Polskę, Bułgarię, Rumunię itd.) i możliwości użycia broni jądrowej. Np. wybór czwartego scenariusza (Starwars Option) powoduje wyeliminowanie z użycia broni atomowej krótkiego, średniego, i dalekiego zasięgu.

Jest to spowodowane wysokim poziomem i masowym zastosowaniem systemów wczesnego ostrzegania i automatów niszczących rakiety przed dotarciem do celu. Do naszej dyspozycji pozostaje broń, atomowa znajdująca się na składzie poszczególnych oddziałów, i wykorzystywana bezpośrednio na polu walki (battlefield).

Ekran gry jest zbudowany z dwóch części: mapy, na której widzimy ruchy wojsk obu stron i szeregu komputerów, pozostających do naszej dyspozycji. Dzięki nim, będziemy wydawali część rozkazów i zbierali informacje dotyczące przebiegu wojny. Pierwszy od lewej monitor informuje nas o populacji ludności cywilnej na terenie walki, drugi - ukazuje stopień radiacji na poszczególnych obszarach po użyciu broni atomowej, trzeci jest związany z planami ataków atomowych, które później zostaną opisane szerzej. Czwarty ekran jest związany z aktywacją specjalnych misji lotniczych, piąty dotyczy rozdziału jednostek lotniczych do poszczególnych zadań, na szóstym możemy zorientować się o dotychczasowym przebiegu wojny, siódmy jest wykorzystywany przy wszelkiego rodzaju działaniach dyplomatycznych. Ósmy podaje nam informacje o ilości posiłków, skierowanych do naszych wojsk w poszczególnych dniach wojny i ostatni, dziewiąty dotyczy opcji gry (ładowanie, zgrywanie, wyjście itp.)



Populacja i stopień promieniowania na danym terenie są zależne od używania broni nuklearnej w tym rejonie. Nie ma to wpływu na grę. Jedynie ilość ofiar cywilnych jest brana pod uwagę w końcowym podliczaniu efektów gry.

Aby uruchomić komputer obsługujący broń atomową, należy podać hasło. Jest nim słowo midnight. Do wyboru mamy dwa rodzaje ataków. Ty atakujesz pierwszy, lub uruchamiasz system szybkiego reagowania. Odpowiedzią na pojedynczy, nuklearny atak przeciwnika jest single strike, a na atak zmasowany multi strike. Jeśli chcemy użyć jakiejkolwiek broni atomowej, to musi-



my znać nazwę danego planu (fireplan). Każdy z nich opisany jest za pomocą następujących danych : nazwa planu, kategoria, typ wybuchu, ilość odpalonych rakiet, głowice bojowe - jakość i liczba.



ROSJA :

May the First - pole walki, eksplozja powietrzna, jedna, pojedyńcza o małej wydajności. Bear Paw - pole walki, atak neutronowy, jedna, pojedyńcza o małej wydajności. Smog Bird - pole walki, eksplozja naziemna, jedna, pojedyńcza o małej wydajności. Red Star - pole walki, eksplozja powietrzna, wiele, wiele o dużej wydajności. Shark Bite - pole walki, atak neutronowy, wiele, wiele o dużej wydajności. Thunder Cloud - pole walki, wybuch naziemny, wiele, wiele o dużej wydajności. Star Burst - krótki zasięg, eksplozja powietrzna, jedna, pojedyńcza o małej wydajności. Broken Glass - krótki zasięg, atak neutronowy, jedna, pojedyńcza o małej wydajności. Hell Fire - krótki zasięg, wybuch naziemny, jedna, pojedyńcza o małej wydajności. Iron Curtain - krótki zasięg, eksplozja powietrzna, cztery, wiele. Atak na cztery najsilniejsze jednostki wroga. Steel Box - krótki zasięg, eksplozja powietrzna, cztery, wiele. Atak na cztery lokalne centra zaopatrzenia. Spear Head - krótki zasięg, eksplozja powietrzna, osiem, wiele. Atak na centra broni atomowej krótkiego zasięgu wroga. War Hammer - krótki zasięg, eksplozja powietrzna, jedna, pojedyncza. Atak na centrum dowodzenia. Steam Roller - krótki zasięg, eksplozja powietrzna, osiem, wiele. Atak na osiem najsilniejszych jednostek przeciwnika. Road Block - średni zasięg, eksplozja powietrzna, osiem, pojedyńcza. Atak na drogi i tory kolejowe. Broken Wing - średni zasięg, eksplozja powietrzna, szesnaście, pojedyńcza. Atak na lotniska. Snowdrift - średni zasięg, atak chemiczny, szesnaście, pojedyńcza. Atak na lotniska. White Tiger - średni zasięg, eksplozja powietrzna, osiem, pojedyńcza. Atak na miasta. Swan Flight - średni zasięg, eksplozja powietrzna, osiem, pojedyńcza o małej wydajności. Sky Rocket - strategiczny, eksplozja powietrzna, jedna, pojedyńcza o dużej wydajności. Mothball - strategiczny, wybuch naziemny, jedna, pojedyńcza o dużej wydajności. Nightmare - lotniczy, atak chemiczny, osiem, pojedyncza. Atak na punkty dowodzenia. Smoke Screen - średni zasięg, atak chemiczny, osiem, wiele. Atak na centra zaopatrzenia. Wheatsheaf - lotniczy, eksplozja powietrzna, szesnaście, pojedyncza. Fog Bank - lotniczy, atak chemiczny, osiem, pojedyncza. Atak na miasta zaopatrzeniowe. Broadsword - średni zasięg, eksplozja powietrzna, osiem, wiele. Atak na centra zaopatrzenia. Tin Tacks - lotniczy, konwencjonalny, szesnaście, pojedyncza. Atak na lotniska. Autumn Mist - lotniczy, atak chemiczny, szesnaście, pojedyncza. Atak na lotniska.

NATO :

Sharp Stick - pole walki, eksplozja powietrzna, jedna, pojedyńcza o małej wydajności. Switchblade - pole walki, atak neutronowy, jedna, pojedyńcza o małej wydajności. Dirty Harry - pole walki, wybuch naziemny, jedna, pojedyńcza o małej wydajności. Karma - strategiczny, eksplozja powietrzna, jedna, wiele o dużej wydajności. Knuckle Duster - pole walki, eksplozja powietrzna, wiele, wiele o dużej wydajności. High Light - pole walki, atak neutronowy, wiele, wiele o dużej wydajności. Headbutt - pole walki, wybuch naziemny, wiele, wiele o dużej wydajności. Little Joe - krótki zasięg, eksplozja powietrzna, jedna, pojedyńcza. White Spot - strategiczny, atak neutronowy, jedna, pojedyńcza. Guillotine - strategiczny, eksplozja powietrzna, jedna. Atak na centrum dowodzenia. Iron Fist - strategiczny, eksplozja powietrzna, osiem. Atak na osiem najmocniejszych jednostek przeciwnika. Keystone - średniego zasięgu, eksplozja powietrzna, osiem, pojedyńcza o dużej wydajności. Mitigation - strategiczny, wybuch naziemny, jedna, pojedyńcza. Fumble Winter - strategiczny, eklsplozja powietrzna, wiele, pełne uderzenie. Paper Bag - lotniczy, konwencjonalny, szesnaście. Atak konwenjonalny na lotniska. Fire Cracker - lotniczy, eksplozja powietrzna, szesnaście, pojedyncza. Grounded - lotniczy, atak chemiczny, szesnaście. Atak chemiczny na lotniska. Braiwash - lotniczy, atak chemiczny, osiem. Atak chemiczny na centra dowodzenia.

Co pewien czas możemy przeprowdzić specjalne misje lotnicze, np. atak na głębokie zaplecze wroga, czy atak chemiczny na jakiś cel. Aby tego dokonać należy uruchomić daną misję na czwartym komputerze.

Podczas trwania konfliktu otrzymujemy posiłki dla lotnictwa strategicznego. Naszym zadaniem jest rozdzielić samoloty między odpowiednie formacje. Dokonujemy tego na ekranie piątym. Formacje są różne, np. panowanie w powietrzu (Air Superiority), przechwytywanie (Interdiction), głębokie uderzenie (Deep Strike), obrona powietrzna (Air Defence), bombardowanie strategiczne (Strategic Bombing) czy lotniczy zwiad (Reconnaisance). Każda z tych grup spełnia określone zadania, bardzo przydatne w czasie walki.

Ekran szósty to informacje o przebiegu starcia.

Na komputerze siódmym prowadzimy negocjacje i rozmowy rozbrojeniowe z przeciwnikiem. Możemy się poddać, zaproponować pokój, czy podpisać pakt o broni atomowej.

W zależności od rodzaju ataków rakietowych wroga na nasze zaplecze gospodarcze zmienia się ilość posiłków kierowanych do armii lądowych. O przewidywanej ilości możemy się dowiedzieć na kolejnym monitorze.

Ostatni ekran obsługuje opcje gry.

Chcę by ten krótki opis zapoznał Was z realiami gry. Nie jest ona wprawdzie szczególnie porywająca czy też zapierająca dech w piersiach. Jest jednak interesująca na tyle, że można spędzić przy niej przyjemnie wolny czas.

Marcin "Emer Curl" Kasprzak

Gry od Czytelników (C-64)



Dostajemy coraz więcej programów autorstwa naszych Czytelników. Większość z nich jest starannie dopracowana. Niestety, tylko niektóre z nich nadają się do publikacji - pozostałe są zwykle za długie! Tak, to jest niestety wadą w przekazywania programów tą drogą (to znaczy na łamach KEBABA). Wszystkim zainteresowanym ukazaniem się Waszych programów w tym magazynie wyjaśniam, że maksymalną długością jest pułap 20 bloków na dysku. Programy większe niż 20 bloków to już koszmarny wydruk na kilkanaście stron w KEBABIE. Może w przyszłości, wraz ze wzrostem objętości, będziemy mogli sobie na to pozwolić ale, niestety, jeszcze nie teraz. Ten los spotkał między innymi moim zdaniem bardzo wszechstronny edytor do projektowania sprite'ów autorstwa Grzegorza Piaskowskiego. Drogi Grzegorzu, trochę ulepszeń i drobnych modyfikacji i jestem przekonany, że uda Ci się "zejść" do odpowiednich "gabarytów" Twojego programu. Szczerze życzę powodzenia!

W tym numerze chciałbym przedstawić dwa programy autorstwa Dariusza Makucha. Na nadesłanym przez niego dysku znalazłem więcej propozycji ale zdecydowałem się tylko na dwie najciekawsze. Są nimi PING-PONG i gra napisana w BASIC - Robot R-23. Ping-Pong to po prostu kolejna wersja przeniesionej ze staruteńkich gier telewizyjnych "odbijanki". W dużym skrócie - dwie kreski (paletki/rakietki?) odbijają do siebie piłkę. Gra ta, mimo zgodnego z oryginałem ubóstwa grafiki, wciąga tak samo i pozwala na podtrzymanie kontaktów towarzyskich (do pełni szczęścia potrzeba bowiem dwóch graczy, każdy z wajchą, przepraszam - z paletką w ręku).

Drugi program to prosta gra zręcznościowo-logiczna (tak to określił sam Autor). Napisano ją w języku BASIC, co można łatwo sprawdzić poprzez wciśnięcie klawisza oznaczonego RUN/STOP. W programie znajduje się instrukcja do gry, która powinna uwolnić naszych Czytelników od czytania moich wypocin na ten temat. Gra ta ma kilka zalet, że tak powiem - edukacyjnych. Głównie chodzi o to, że została napisana w języku BASIC, przez co można ją sobie dokładnie obejrzeć, przeanalizować lub nawet twórczo zmodyfikować. Do tej ostatniej możliwości gorąco razem z Autorem namawiam. W tym miejscu "na miejscu" byłby cytat z listu, gdzie w pewnym jego miejscu Autor sugerował wprowadzenie drobnych poprawek w linii o numerze 130. Znajduję się tam instrukcja ON P GOTO.... Wystarczy dopisać numery nowych, napisanych przez nas linii programu. A co te nowe linie mogą zawierać? Ni mniej, ni więcej, tylko kształty nowych ekranów gry! Należy jednak pamiętać o tym, żeby ostatnia z liczb wynosiła 395, gdyż skok do tej linii pozwala na prawidłowe zakończenie gry.

Pierwszy z programów został zamieszczony w postaci listingu spod KOREKTORA, co czyni go także możliwym do wpisania przy użyciu dowolnego debuggera. Drugi, z racji swej większej objętości, został wydukowany w KEBABIE w postaci danych do programu INPUTER. Sam zaś Inputer opublikowany był w 5 numerze KEBABA.

Paweł Sołtysiński

Grupa **"DEFENCE"** (C64) **poszukuje** członków wszelkich specjalności oraz kontaktów. Napisz! Sławomir Rogalski, ul. Makarskiego 47/3, 49-300 Brzeg.

Kupię tanio programy na C64, koperta + znaczek (taśma, dysk). Romuald Rodzik, ul. Sienkiewicza 12/12, 24-100 Puławy.

Nawiążę kontakt z osobami uczącymi się asemblera. Posiadam A500. Damian Surdyn, ul. Piłkowska 7/4, 02-590 Warszawa.

Grupa **"VADER"** (C64) **nawiąże** kontakty. Poszukujemy nowych członków, szczególnie koderów.
Piotr Janowski, ul. Lelewela 7/23, 78-200 Białogard.

Sprzedam stację 3.5" do Amigi oraz rozszerzenie 1.8 Mb. Bogdan Marczuk, ul. Wielka 41/7, 53-338 Wrocław, tel. 678-195.

FIATA 126p rok 1980 na drukarkę CITIZEN SWIFT 9SX lub CD-ROM A570 + compakty lub CDTV **zamienię**. Henryk Wójcik, ul. Piłsudskiego 78/11, 58-301 Wałbrzych.

ACTION VISION **poszukuje** nowych członków wszelkich specjalności. Piszcie na adres: Krzysztof (CHRIS) Gil ul. Zaściańska 149/2, 15-546 Białystok.

Użytkownicy C64 i swapperzy wszelkich grup łączcie się. Grupa **"FATUM"** **piekielnie poszukuje** muzyków!!!. Waldek Jedwabnik, ul. Sikorskiego 41/6, 11-200 Bartoszyce.

C64 + magnetofon BLACK BOX V.2 HELP PL SUPER-EXPANDER PLUS oraz 24 kasety (ok. 900 programów) joystick i literatura. Całość 1.8 mln. Leszek Mankiewicz, ul. Wojska Polskiego 8/12, 72-010 Police.

Sprzedam C64 + stacja 1541 + dodatki (Final III, Black Box, dyski, kasety). Marek Juszkiewicz, ul. Łucznicza 66/39, 71-577 Szczecin.

Mapa pamięci C64. Cena 60 tys. - płatne przy odbiorze. (Zawsze aktualne!). Tomasz Filipowicz, Dąbrowica 26/5, 58-500 Jelenia Góra.

Wymiana gier i użytków C64 (disk only). **Kupię** schemat samplera do C64 - pilne! **Wymienię** użytki - Amiga. Jędrzej Chmielewski, ul. Kąkolowa 7/33, 85-811 Bydgoszcz.

Kupię pilnie TURBO ASSEMBLER V.5.1. Tomasz Pupiec, ul. Bukowa 21/89, 20-353 Lublin, tel. 435-47.

Kupię książkę MC 68000, Piotr Forneila, ul. Klemensiewicza 10/4, 70-028 Szczecin

Mapa pamięci C64. Za 40 tys. zł otrzymujesz pełny opis wszystkich lokacji pamięci na nośniku magnetycznym. Andrzej Gruszczyński, ul. Zielonogórska 67/3, 66-016 Czerwieńsk.

Kupię stację dysków 1541 II, odpiszę na wszystkie listy. Jarosław Syrek, ul. Roweckiego 14 B/5, Szczecin.

Pomóżcie mi w Asemblerze! Mogę wstąpić do grupy (Amiga). Zdolności graficzne Piotr Gościński, ul. Piastowska 2/15, 11-400 Kętrzyn, tel. 63-76.

Sprzedam ponad 1000 programów (kaseta) na C64. Katalog: koperta+znaczek. Tomasz Dąbkowski, Racław 15, 66-432 Baczyna.

Sprzedam nowości do C64/128 (dysk). Katalog gratis. Dema oraz magazyny wydawane przez polskie grupy **rozpowszechniam** bezpłatnie. Mariusz Wośko, Piłsudskiego 18/30, 48-303 Nysa.

Grupa **"GEDEON"** (C64) **poszukuje** koderów, grafików i muzyków. Wszelkie prace, pytania itp kierować na adres: Adamiec/Gedeon - Adam Zelent, ul. Sudecka 2/10, 48-300 Nysa.

Sprzedam: A 2000B + A 2092 SCSI (20 MB) = 9.9 mln zł; MEMORYMIKSTER 2/8 MB RAM = 2.9 mln zł,!!! komplet = 12.5 mln zł. Robert Tchórzewski, ul. Pomorska 18d/33, Gdańsk, tel. (058)579-774 (19.00-22.00.

Amiga! **Programy**. Co czwarty program gratis! Katalog gratis (koperta+znaczek. Wojciech Turczyn, ul. Jeziorna 1/17, 10-153 Olsztyn.

Kupię A500, na raty, warunki spłat + opis sprzętu + cena. M. Książak, ul. Prosta 8b/4, 07-200 Wyszków.

THE KING - mocny **kontakt** dla posiadaczy C-64 i A500 (nowości). Paweł Witek, ul. Karłowicza 45/55, 58-506 Jelenia Góra.

C-64 II z osprzętem (400 gier) cena 1.8 mln zł. Dariusz Węgiel, Nawojowa Góra 321, 32-065 Krzeszowice.

Sprzedam C64II, magnetofon, joystick, Black Box, 200 programów lub **zamienię** na Amigę. Piotr Lewandowski,

Wpłata dotyczy:

Wpłata dotyczy:

Wpłata dotyczy:

Sprzedam: Amigę 500 1MB, przełącznik Kickstartów 1.3 - 2.0, bootselektor, Chip-Fast, modulator, joystick, ok. 100 gier i programów użytkowych (także oryginalne). Cena 7 mln. zł. Zbigniew Kitowski, ul. Rolna 10, 78-200 Białogard

Oś. Zachód B/1/H/12, 73-110 Stargard.

Sprzedaż oraz wymiana gier i użytków. Katalog (znaczek + koperta). Sebastian Wesołowski, ul. Lotników 1/21C, 78-520 Złocieniec.

THE KING **wymieni** programy Amiga 500 i C64 (dysk). Nie ma dysku - nie ma odpowiedzi. Paweł Witek, ul. Karłowicza 45/55, 58-506 Jelenia Góra.

Sprzedam: Amiga 500 1MB, monitor 1084S, 200 dysków i literaturę. Cena 10 mln zł. Marcin Biszczad, ul. B. Prusa 61, 22-100 Chełm, tel. 522-76

Polsko-Austriacki klub AMIGA. **Darmowa wymiana** gier. Szczegóły: koperta + znaczek. Robert Maślańczyk, Taglieberstrase 1/10, 1230 Wien, Austria.

Sprzedam - A500 (V1.3, angielska - 5 mln, modulator- 3 mln, rozszerzenie 1 MB - 450 tys. Piotr Bankowski, ul. Pogorska 10B/10, 32-500 Chrzanów, tel. (035)378-62.

Niedawno powstała grupa **F.O.I. poszukuje** członków wszelkich specjalności i umiejętności. Kontakt : Marcin Lewandowski, ul. J. Korczaka 35, 87-300 Brodnica.

Sprzedam syntezator mowy - przystawka do C64 w zestawie do złożenia. Stanisław Ciszewski, ul. Zdrojowa 43, 57-320 Polanica Zdrój.

Kupię mapę pamięci C64 - propozycje na kartkach pocztowych. Grupa **POLARED nawiąże** kontakt z grupami uczącymi się Assemblera. Sławomir Stolarczyk, ul. Sobięcińska 4, 58-370 Boguszów-Gorce.

C64 - gry i użytki oraz opisy. Informacje: koperta + znaczek. Marcin Listowski, ul. Sobieskiego 17/8, 76-200 Słupsk.

Katalogi u. scal. TTL, MOS, UPC, pamięci, AD/DA, liniowe - firm zach. - **sprzedam**. Oferty tylko listowne. Szczecin, ul. Derdowskiego 30/6.

SPRZEDASZ/ZAMIENISZ C64 na Amigę kupując poradnik - 40 tys. (koszty druku) + duża koperta zwrotna i znaczek. Sebastian Zabrzeński, ul. Krucza 4/2, 59-300 Lubin.

Sprzedam magnetofon CA 12 do Atari - nieużywany - 400 tys. Krzysztof Łączyński, ul. Padlewskiego 11/27, 09-402 Płock, tel. 640-884.

Nawiążę kontakt z posiadaczami CB-radia połączonego do Commodore C64. Henryk Stefanowicz, ul Adama 13/16, 40-467 Katowice.

Grupa **DREAM F.** (Amiga) **nawiąże** kontakty. Poszukujemy grafików (dobrych). Piszcie na adres: Dawid Foremski, Oś. 700-lecia 13/38, 34-300 Żywiec.

Sprzedam C64, stację 1541-II, magnetofon, 25 kaset, 30 dyskietek + pudełko, cartridge: Bis Plus, Final III - cena 3 mln. Łukasz Telichowski, ul. Chmielna 1, 64-300 Borzyce Kościelne, tel. 113-16.

Grupa ATHEIST **poszukuje** członków wszelkich specjalności. Nawiążemy kontakt. Jacek Wojtan, ul. Koszarowa 13/14, 40-068 Katowice.

FATUM przyjmie grafików, mózgowców (pomysły), muzyków i swapperów - C64. Wojciech Chmiel, ul. Mickiewicza 21/2, 43-300 Bielsko-Biała.

"SAF" (Klub Użytkowników Amigi) - grafika, Amos, dema, wymiana programów użytkowych i doświadczeń. Tomasz Frontczak, ul. Wyszyńskiego 90A, 62-650 Kłodawa.

Sprzedam programy do C64/128, nowości to moja specjalność, 1200 pozycji dyskowych. Katalog gratis. Mariusz Wośko, ul. Piłsudskiego 18/30, 48-303 Nysa.

Sprzedam C64, stacja 1541 II, 60 dysków+pudełko, cartridge X, Action V7.3, magnetofon, kasety z grami, joystick. Cena ok. 4 mln. Wojciech Gołębiewski, ul. Kolberga 2D/1, 81-881 Sopot

Poszukuję opisu mikroswitchów do STAR NL-10C. Wymienię gry i użytki na C64, Amigę. **Kupię** schemat samplera do C64. Jędrzej Chmielewski, ul. Kąkolowa 7/33, 85-811 Bydgoszcz.

Wyprzedaż gier do C64. Cena dysku 15 tys. Katalog: koperta + znaczek. Mariusz Listowski, ul. Sobieskiego 17/8, 76-200 Słupsk.

ESTATE (C64) **poszukuje** kontaktów oraz członków wszelkich specjalności (szczególnie koderów). Wiesław Walczyk, ul. Tuwima 6/38, 47-225 K-Koźle.

Sprzedam tanio C64 na gwarancji + magnetofon + joystick + 8 kaset + Black Box V4 za 1.8 mln. Janusz Zawistowski, ul. Słowiańska 21B/17, 78-400 Szczecinek.

Sprzedam Amigę 500 1.3 1MB + modulator + joystick + 40 dysków + literatura. Cena 5.9 mln. Seweryn Niemiec, ul. E.Gierczak 43/12, Szczecin, tel. 600-910.

Kurs "Jak napisać własne intro/demo" - wymagana znajomość podstaw assemblera (C64). Tomasz Przybylski, ul. Waryńskiego 10A/1, 47-223 Kędzierzyn.

Grupa **"ANTI"** (C64) **poszukuje** kontaktów i (bardzo) zdolnych ludzi chętnych do wstąpienia. AG/ANTI, ul. Lewakowskiego 9/8, 38-400 Krosno.

Wymienię oprogramowanie (dema, gry, magazyny) na C64 - dysk, 100% odpowiedzi. Jarosław Kothe, ul. 1-go Maja 49, 64-100 Leszno.

Sprzedaż oprogramowania na Amigę. Duży wybór (ponad 600 pr.). Niedrogo! Grzegorz Morzewski, ul. Szeligowskiego 4/64, Konin (Zatorze), tel. 440-208.

Kupię oryginalną wersję TURBO ASSEMBLERA V5.1 taśma lub cartridge Kazimierz Szymański ul. Wagowa 40/7 42-540 Sosnowiec

Sprzedam: Amigę 2000 + monitor, drukarka Star LC-20. Cena do uzgodn. Jacek Bandzmer, ul. Grabowskiego 11/18, 80-809 Gdańsk, tel. 329-207.

Sprzedam Amigę 500+, 2MB, gwarancja - 12'93, pokrywę, mysz - 6.8 mln. Adam Sokołowski, Os. Dolne Miasto 14/26, 78-600 Wałcz, tel. 62-26.

MARTINI zaprasza szukających kontaktu w celu wymiany oprogramowania na C64/128 - dysk (gry, użytki, dema...). 111% procent odpowiedzi. Marcin Dajewski, ul. Spacerowa10, 64-100 Leszno

Zamienię fotoaparat Kijew-19, teleobiektyw, lampę na stację do Commodore 64. Krzysztof Wysocki, ul. Poznańska 4, 99-400 Łowicz.

Sprzedam monitor Commodore 1084S, cena ok. 3.5 mln Janusz Jura, ul. Mazowiecka 30A/13, 81-862 Sopot, tel. 519-209.

Sprzedam lub **wymienię** gry i użytki na C64. Informacje (koperta + znaczek). Sebastian Wesołowski, ul. Lotników 1/21C, 78-520 Złocieniec.

Sprzedaż oprogramowania do Amigi. Co tydzień nowości. Katalog gratis. Przemysław Mikosz, ul. Buczka 27/12, 43-300 Bielsko-Biała, tel. 495-37.

Sprzedam gry całodyskowe (C64). Katalog - koperta + znaczek. Bartosz Kiełczewski, ul. Małeckiego 7/13, 10-293 Olsztyn.

Zamienię organy CA-301+ automat perkusyjny na Amigę 500. Dariusz Gawerski, ul. Sportowa 20, 11-200 Bartoszyce.

Kupię stację do C64 - gotówka lub gotówka + organy MC-3DX. Jacek Antczak, ul. Kilińskiego 3a, 62-860 Opatówek.

Listingi

Przykład 1 do Amosa

```
Dim LICZBY(5),LOSY(5)
Global LICZBY(),LOSY()
SETUP
STAR:
PLANSZA
ZAKLADY
LOSUJ
Goto STAR
Procedure SETUP
   Screen Open 0,320,256,8,LORES
   Colour 1,$FFF
   Curs Off : Hide
   Paper 0 : Ink 1 : Pen 1
   Cls
End Proc
Procedure PLANSZA
   Cls
   Ink 5,0
   Screen Open 1,50,20,2,LORES
   Print At(0,0);"Lotto"
   Zoom 1,0,0,40,8 To 0,55,10,250,50
   Screen Close 1
   For F=3 To 10
      Draw 5,F*18 To 129,F*18
      Draw F*18-50,54 To F*18-50,180
      Draw 5+160,F*18 To 129+160,F*18
      Draw F*18+110,54 To F*18+110,180
   Next
   Ink 1,0
   Gr Writing 0
   For H=0 To 6
      For F=1 To 7
         F$=Str$(F+H*7)
         Text 18*H-2,F*18+49,F$
         Text 18*H+158,F*18+49,F$
      Next
   Next
End Proc
Procedure ZAKLADY
   Restore KOLEJKA
   For F=0 To 5
      Read KOLEJKA$
      NIETAK:
      Print At(0,24);Space$(62)+Cup$+Cup$
      Input "Podaj "+KOLEJKA$+" liczbe: ";A
      If A<1 or A>49 Then Boom : Goto NIETAK
      For P=0 To 5
         If LICZBY(P)-A Then Boom : Goto NIETAK
      Next
      LICZBY(F)=A
      MALUJ[A,0]
   Next
   Curs Off
   Print At(0,24);"Gotowy do losowania?"+Space$(10)
   Wait Key
   Print At(0,24);Space$(30)
   KOLEJKA:
   Data "pierwsza","druga","trzecia","czwarta","piata","szosta"
End Proc
Procedure MALUJ[NUMER,PLANSZA]
   Ink 4,0
   If NUMER<8 Then YOFF=NUMER*18 : XOFF=0
   If NUMER=>8 and NUMER<15 Then YOFF=(NUMER-7)*18 : XOFF=18
   If NUMER=>15 and NUMER<22 Then YOFF=(NUMER-14)*18 : XOFF=2*18
   If NUMER=>22 and NUMER<29 Then YOFF=(NUMER-21)*18 : XOFF=3*18
   If NUMER=>29 and NUMER<36 Then YOFF=(NUMER-28)*18 : XOFF=4*18
   If NUMER=>36 and NUMER<43 Then YOFF=(NUMER-35)*18 : XOFF=5*18
   If NUMER=>43 Then YOFF=(NUMER-42)*18 : XOFF=6*18
   Bar 5+PLANSZA*160+XOFF,YOFF+37 To 21+PLANSZA*160+XOFF,YOFF+53
End Proc
Procedure LOSUJ
   For F=0 To 5
      Randomize Timer
      ZLE:
      LOS=Rnd(48)+1
```

```
      For P=0 To 5
         If LOSY(P)=LOS Then Goto ZLE
      Next
      Wait 50
      Bell 50
      LOSY(F)=LOS
      MALUJ[LOSY(F),1]
   Next
   TRAF=0
   For F=0 To 5
      For T=0 To 5
         If LICZBY(F)=LOSY(T) Then Inc TRAF
      Next
   Next
   Print At(0,25);"Trafiles";TRAF;" raz(y)"
   Print "Grasz jeszcze raz? (t/n)"
   Do
      I$=Inkey$
      If Lower$(I$)="t" Then Pop Proc
      If Lower$(I$)="n" Then End
   Loop
End Proc
Procedure KASUJDANE
   For F=0 To 5
      LOSY(F)=0 : LICZBY(F)=0
   Next
End Proc
```

Przykład 2 do Amosa

```
Dim S#(180)
Global S#()
SETUP
FUNKCJA
Procedure SETUP
   Screen Open 0,640,256,2,Hires
   Colour 1,$FFF
   Curs Off : Hide
   Paper 0 : Ink 1 : Pen 1
   Cls
End Proc
Procedure FUNKCJA
   Set Rainbow 0,0,100,"","(1,2,7)(1,-2,7)",""
   Set Rainbow 1,0,100,"","(1,2,7)(1,-2,7)",""
   Degree
   Rem Definiowanie funkcji :
   Def Fn F(X)=2*(-Sin(3*X)*Cos(X))
   For X=0 To 180
      S#(X)= Fn F(X)*22+100 : Rem Liczenie i tablico-
                                 wanie wyniku w S#()
   Next
   Do
      Cls
      Locate 0,10
      Centre "Procedura przykladowa v0.1"
      Plot 0,S#(0)+42
      For W=0 To 3
         For X=0 To 180
            If Inkey$<>"" Then Exit 3
            Draw To X+180*W,S#(X)+42
            Wait Vbl
            Rainbow 0,0,S#(X),15
            Rainbow 1,0,300-S#(X),15
         Next
      Next
   Loop
   Cls
   A=S#(X) : B=300-A
   Repeat
      If A>50 Then Dec A
      If B<270 Then Inc B
      Wait Vbl
      Rainbow 0,0,A,15
      Rainbow 1,0,B,15
   Until A=50 and B=270
   Wait Key
End Proc
```

```
"PING                    "              $0801-$0C00
-------------------
:0801 0B 08 0A 00 9E 32 30 36 (84)
:0809 31 00 00 00 A9 00 8D 20 (6A)
:0811 D0 A9 06 8D 21 D0 A9 07 (DD)
:0819 8D 86 02 A9 93 20 D2 FF (B9)
:0821 4C 33 08 50 55 4E 4B 54 (5D)
:0829 59 4D 45 43 5A 4C 45 57 (24)
:0831 45 4C A9 40 A2 28 9D FF (6E)
:0839 03 9D CF 06 CA D0 F7 A2 (A6)
:0841 06 BD 23 08 9D 25 07 CA (C2)
:0849 D0 F7 A2 04 BD 29 08 9D (CC)
:0851 3A 07 CA D0 F7 A2 05 BD (E9)
:0859 2D 08 9D 58 07 CA D0 F7 (1C)
:0861 A9 30 8D 77 07 8D 79 07 (ED)
:0869 8D 8B 07 8D 8D 07 A9 31 (6F)
:0871 8D AB 07 A9 3A 8D 78 07 (05)
:0879 8D 8C 07 A2 40 A9 00 9D (E1)
:0881 BF 02 CA D0 FA A9 FF A2 (CB)
:0889 00 9D C0 02 E8 E8 E8 E0 (63)
:0891 3F D0 F6 A9 01 8D 27 D0 (E2)
:0899 A9 0D 8D 28 D0 8D 29 D0 (A8)
:08A1 A9 0B 8D F9 07 8D FA 07 (72)
:08A9 A9 2F 8D F8 07 A9 24 8D (BC)
:08B1 02 D0 A9 40 8D 04 D0 A9 (27)
:08B9 04 8D 10 D0 A9 72 8D 03 (3B)
:08C1 D0 8D 05 D0 A9 07 8D 15 (FC)
:08C9 D0 A2 03 8E AA 02 CA CA (5A)
:08D1 CA 8E AD 02 8E AE 02 4C (16)
:08D9 47 09 AD AE 02 D0 1E A2 (C5)
:08E1 01 8E A9 02 8E AE 02 CA (41)
:08E9 8E AB 02 20 1B 09 20 31 (80)
:08F1 09 AD 10 D0 29 FE 8D 10 (E8)
:08F9 D0 20 26 09 60 A9 00 8D (E5)
:0901 A9 02 8D AB 02 8D AE 02 (34)
:0909 20 1B 09 AD 10 D0 09 01 (AE)
:0911 8D 10 D0 20 3C 09 20 26 (29)
:0919 09 60 AD 12 D0 29 07 09 (B9)
:0921 02 8D AC 02 60 AD 12 D0 (3E)
:0929 09 80 29 9F 8D 01 D0 60 (A9)
:0931 AD 12 D0 29 3F 09 22 8D (E6)
:0939 00 D0 60 AD 12 D0 29 3F (07)
:0941 09 12 8D 00 D0 60 A9 00 (0D)
:0949 85 02 78 AD 14 03 AE 15 (D7)
:0951 03 8D A7 02 8E A8 02 A9 (80)
:0959 A1 A2 0A 8D 14 03 8E 15 (99)
:0961 03 58 AD 15 D0 29 FE 8D (D8)
:0969 15 D0 A9 00 8D 38 0A A9 (C1)
:0971 01 2C 00 DC F0 05 2C 01 (4D)
:0979 DC D0 03 20 57 0A A9 02 (25)
:0981 2C 00 DC F0 05 2C 01 DC (12)
:0989 D0 03 20 65 0A A9 10 2C (54)
:0991 00 DC F0 0B 2C 01 DC F0 (B4)
:0999 06 20 37 0A 4C 70 09 A9 (58)
:09A1 01 0D 15 D0 8D 15 D0 A9 (7B)
:09A9 01 8D B0 02 8D B1 02 20 (DA)
:09B1 42 0A 20 DB 08 A9 01 85 (29)
:09B9 02 AD B0 02 F0 E9 AD B1 (9F)
:09C1 02 D0 F6 20 42 0A 4C 0D (D0)
:09C9 08 A9 01 2C 00 DC D0 0A (07)
:09D1 AC AA 02 88 20 0B 0A 88 (68)
:09D9 D0 FA 2C 01 DC D0 0A AC (00)
:09E1 AA 02 88 20 16 0A 88 D0 (92)
:09E9 FA A9 02 2C 00 DC D0 0A (1C)
:09F1 AC AA 02 88 20 21 0A 88 (0C)
:09F9 D0 FA 2C 01 DC D0 0A AC (20)
:0A01 AA 02 88 20 2C 0A 88 D0 (21)
:0A09 FA 60 AE 03 D0 E0 38 F0 (3B)
:0A11 03 CE 03 D0 60 AE 05 D0 (9A)
:0A19 E0 38 F0 03 CE 05 D0 60 (23)
:0A21 AE 03 D0 E0 AF F0 03 EE (5F)
:0A29 03 D0 60 AE 05 D0 E0 AF (3F)
:0A31 F0 03 EE 05 D0 60 A2 F0 (4D)
:0A39 A0 00 C8 D0 FD E8 D0 F8 (4C)
:0A41 60 A9 00 85 02 A9 04 2C (8D)
:0A49 00 DC F0 05 2C 01 DC D0 (55)
:0A51 F6 A9 01 85 02 60 AE AA (16)
:0A59 02 E0 06 F0 06 EE AA 02 (5F)
:0A61 EE AB 07 60 AE AA 02 E0 (B4)
:0A69 03 F0 06 CE AA 02 CE AB (F8)
:0A71 07 60 CE AD 02 D0 28 AD (CA)
:0A79 AC 02 8D AD 02 AD AB 02 (63)
:0A81 F0 0B AD 01 D0 C9 BA F0 (F8)
:0A89 0F EE 01 D0 60 AD 01 D0 (36)
:0A91 C9 32 F0 08 CE 01 D0 60 (74)
:0A99 CE AB 02 60 EE AB 02 60 (03)
:0AA1 A5 02 D0 03 4C FF 0A 20 (8C)
:0AA9 CA 09 AE AA 02 20 73 0A (80)
:0AB1 CA D0 FA AD A9 02 F0 1C (90)
:0AB9 18 AD AA 02 6D 00 D0 8D (74)
:0AC1 00 D0 90 0D AD 10 D0 09 (08)
:0AC9 01 8D 10 D0 A9 00 8D 00 (86)
:0AD1 D0 4C EE 0A 38 AD 00 D0 (DB)
:0AD9 ED AA 02 8D 00 D0 B0 0D (76)
:0AE1 AD 10 D0 29 FE 8D 10 D0 (00)
:0AE9 A9 FF 8D 00 D0 A9 01 2C (AE)
:0AF1 10 D0 F0 0D AD 00 D0 C9 (08)
:0AF9 37 B0 58 4C FF 0A 4C 31 (A5)
:0B01 EA AD 00 D0 C9 28 90 03 (75)
:0B09 4C FF 0A AD 01 D0 38 ED (05)
:0B11 03 D0 B0 04 49 FF 69 01 (2D)
:0B19 C9 0E B0 08 A9 01 8D A9 (AF)
:0B21 02 4C FF 0A EE 79 07 AD (00)
:0B29 79 07 C9 38 F0 08 A9 00 (75)
:0B31 8D B0 02 4C 1D 0B A9 30 (51)
:0B39 8D 77 07 8D 79 07 EE 8D (79)
:0B41 07 AD 8D 07 C9 33 F0 03 (37)
:0B49 4C 2F 0B A9 00 8D B1 02 (F8)
:0B51 4C 1D 0B AD 01 D0 38 ED (8C)
:0B59 05 D0 B0 04 49 FF 69 01 (77)
:0B61 C9 0E B0 08 A9 00 8D A9 (F1)
:0B69 02 4C FF 0A EE 77 07 AD (3C)
:0B71 77 07 C9 38 F0 08 A9 00 (BB)
:0B79 8D B0 02 4C 65 0B A9 30 (01)
:0B81 8D 77 07 8D 79 07 EE 8B (B1)
:0B89 07 AD 8B 07 C9 33 F0 03 (79)
:0B91 4C 77 0B A9 00 8D B1 02 (D0)
:0B99 4C 65 0B 00 00 00 00 00 (DB)
:0BA1 00 00 00 00 00 00 00 00 (AC)
:0BA9 00 00 00 00 00 00 00 00 (B4)
:0BB1 00 00 00 00 00 00 00 00 (BC)
:0BB9 00 00 00 00 00 00 00 00 (C4)
:0BC1 00 00 00 00 00 00 00 00 (CC)
:0BC9 00 00 00 00 00 00 00 00 (D4)
:0BD1 00 3E 00 00 7F 00 00 FF (CB)
:0BD9 80 00 FF 80 00 FF 80 00 (DB)
:0BE1 7F 00 00 3E 00 00 00 00 (63)
:0BE9 00 00 00 00 00 00 00 00 (F4)
:0BF1 00 00 00 00 00 00 00 00 (FC)
```

```
"ROBOT R23          "              $0801-$16F6
-------------------
0801: 2Mw0 09U! c3kV 0a80 u8&1 LiU8 3e
0813: DvE0 UayM 1HQw 2pQQ 0YHg Xq80 44
0825: xGDa CG80 jaU3 3LY4 6!El 87g3 15
0837: A4Mw t0eM 1aA1 Q2Uw d0f9 0%05 dc
0849: 66A2 Q28w d0f9 0%05 66A5 Q1rE 31
085b: 83g3 !gvM 1hxF 2d09 EwAw d0eA ae
086d: FUiF xqyB %EnT Fv#5 #21V 0WnT b7
087f: xvWB #8n% 87g3 I3Aw t0eM 1W86 d4
0891: 83g3 40!y 2i0Q 0NxF g902 VGuw 8c
08a3: 0EiE xq&& FvNB FEnT FvRB FUnU 1d
08b5: Fvvg 0IrU NLsw x0dc 004w t0eM 52
08c7: dGA3 xqww t0eM 6G88 Gi2w 08k2 f8
08d9: x0cw d0c& pga5 FGmD pge5 FV2Z 81
08eb: 87g3 I0qF 0a85 Qd#F 8a01 EwPg b1
08fd: Si1Q 0X09 87g3 Gg1F 1d2# 87g3 ed
090f: I0ww d0c& qgrg Iq88 83g3 QaGF 17
0921: 08mD 1LLM 2iEC FYHg ZG82 &a0Y f1
0933: ba1U z6g3 ia00 IvWA %J02 NL%6 61
0945: %I3D Y0wU aEnX q4MY 0Wj% M0vg 25
0957: YGAT xg5& j0Q8 1LLM Q60U FvvB 64
0969: G8nT I0b6 #aiE Y1KB %3zB G8nY e0
097b: I0b6 %r7T yeUx Q97Y Cd3R NaDg c5
098d: 0m2N ZYrZ NLz6 Gh3E 0bSP 2pTE 7c
099f: 1#zg Z#WM 0#WP 0YrW QeSy 0AM0 e9
09b1: 0gg0 2Mxe hFU! c3&N #CBx 85L% 1d
09c3: !BT3 zy7g E2yn VNzn Kle2 HLY3 f9
09d5: yewh u8gN #wuZ PgyY hcEg #9GV 0c
09e7: Rgyp #8n4 Exbe i0ze iMyV 11LF 40
09f9: %Bc& pMPM bsIs f9Mf 3h01 0Ngf df
0a0b: 4yIw a0cF cjAU ei18 #cYW 0gY6 5e
0a1d: 8081 0xA7 0vDj tqAu u&WU !hV! b7
0a2f: DLzg YHCE vi3U Jj61 bm4# &0jS a1
0a41: j2uc g%3n i2D0 cgZE aj#5 3Uge c0
0a53: 9BZC f0xa 9ywj m&3N FgUq zs&f 31
0a65: QejM J3HJ sKU1 23DZ 8kN9 VyQ8 96
0a77: e!Vw Nw7e VgfK 6djC 0u&L Q0bC 66
0a89: cb4L &fM3 !qS8 y%JA MLp! Fz3M 32
0a9b: 3sAe Y1j9 AL0F Vq%1 2FMw mqpc 82
0aad: HGvT RcHm 2WFq #zrr PaAe QfiY d7
0abf: BL0& avPM 47Kx xvDc 8fs0 J3TW 20
0ad1: YaWF AJ3m %a0Q 48Se 0i03 0p4J 62
0ae3: 66DE j%zC bCiX Y%2e GqmW 0h!F c1
0af5: 0231 rA05 BM6E Gy2W %WD0 EA6w c3
0b07: 0!2Z %WQZ 9sb% %aQY 0UnX HzU3 ce
0b19: H3Y3 GvIw SfYw MU49 %PKU UHz7 82
0b2b: 0j6u j6jy WGB0 xsmB NsB0 YfFw 4f
0b3d: 7%ML Re8y c%4! 394C &da7 zl2n 9f
0b4f: Yldj 85dg NQZ5 jWMw y&d9 3ZGy 6d
0b61: vXk# tM05 xe&q bhK6 a0vX #8up 4a
0b73: q8A1 xW8w 5fW7 &3CM 04HC muAa f2
0b85: F#MG NXCP 1ubz 8wfY 3eX% 3erz 78
0b97: u4nM Q0yt 807E U43g ZkMi 2fpf e5
0ba9: 2eEn L%0e 0kPm bsDU 3uUK 3n&M 2c
0bbb: vK0c 3KYR 6gw5 0num 5hXC 7we3 0c
0bcd: RcTz Mr80 yvjj 4WTc j0wf zg5V 84
0bdf: 7mtt gF17 ePE8 dTg8 4dyL s1F& 70
0bf1: 4ECs 215L fHEl EBuk iiUm 7z34 9e
0c03: 21bK bpVH 81PJ 45s# w3a2 IKM8 e8
0c15: 4Twr MCkr wuv3 k2hg ZMS1 7wwV 58
0c27: 77A2 1V!8 !FGd z%0A 13Q7 ExAu 68
0c39: 1h0E 4y0g d!UV 47iq gA89 P3c4 d8
0c4b: 06s9 7#U5 g5ZV t205 7KB0 48Q9 a0
0c5d: 8bwx I0ka 8nu1 5D0n !7i6 28aP 6b
0c6f: 2i7h 3hkF sdq1 IiEc 4zxR w840 c3
0c81: SgD# 11am 7wgV a7gn %PJ2 E7q3 0a
0c93: ty0# %wAz QNku 1Exb siFT wDi2 13
0ca5: eDkU 471Q w8w1 uiwl %CY5 ds21 c2
0cb7: 6kpf kGKV gSnk g37p dwEJ yIX1 4f
0cc9: g8mF pLyM MAka cHsZ Pf1m 2zdZ 7a
0cdb: 3!YY bmwa ddBu xT51 tvKj 07Ma d7
0ced: d%lR K5T7 sEtN jHe7 2zPZ FsTn dc
0cff: IgF1 tE4C &Spy el2Y wkU! pnUU 02
0d11: a17m krga gKk3 Hglg lztn iA52 c6
0d23: c4Wi iQNl AR&w Dy&5 b16f apl2 db
0d35: kwIw %MF3 Jw5E p532 M2Fk Wxwp d6
0d47: hcAh hbBn 8%dB !S0b hcsr sbLh c8
0d59: JBht Gup0 2QmR JNag STxX ub7M 00
0d6b: mkfs hixB qMJ6 S2Ys QT1m Iv0e 82
0d7d: kXg7 gPb5 yFwb hXQf 1r0p dA57 87
0d8f: jAlk bHU% MwJ8 Wg6s Szxb mAa8 14
0da1: kWhc iyVa bih2 dl2m A5EU j44w 03
0db3: hKQb imA2 e29a kQeT kBF5 gF%e 6f
0dc5: h2!2 mAZc c&18 jAZn G9Qr 34H& a0
0dd7: 1T2# JgkJ jdkK llt1 mwav E6N7 9e
0de9: h5CS m&xa h5GQ 0kAc iXMj Gddj 20
0dfb: ly0u gMEk q3Ne ix0X 9MT# 0A54 48
0e0d: 86Rd ii1q hQDg FYNj mH9x 34N5 36
0e1f: sN5y cmmJ P4aq 6t3c 16Ac kdAb 10
0e31: pE4c lvwe y7cS c7Kq ywNA riii e0
0e43: BFwc pmRX j6gU REGw 36A0 k0Uf 1e
0e55: 0aEc rK4m Mw1T 0bsc sMUL 5Acm 21
0e67: dw3a 37wP z!51 kR&W G3W! cdgc eb
0e79: vtkf I3fS 0eUc ww2h 850w yhU4 d0
0e8b: wZj8 WY7% Qbfk Sbfe Vw0a 3&u0 17
0e9d: %jhQ R5Gp mgl3 DADS 3M16 3&!Q 7e
0eaf: sR2r bax2 CkY5 jpV2 BBDV AMFj a0
0ec1: Ckk5 kIZ3 ghkG 7hQd RAKp j01l ca
0ed3: DAem mFE% 8w1J 3p63 1U&y AgUd e4
0ee5: 7vdy gVBq 1k6m kNMJ 1t8v gM2m 8a
0ef7: 3p&F DA1r zxIL Hf2K UpQY H3cd b8
0f09: CNrX M3IE e3w0 M0Sw TH%0 az8V aa
0f1b: dBd3 b3s0 UMSB YPVv wZvM W2DF ac
0f2d: TZif 4v4d Gw3s LYld a37C F01Q 3a
0f3f: H!Gf SBzx 7Gme J01B gNUR &j4U 87
0f51: Jg6b GEAN eg0M uXC4 hkeP p0D9 06
0f63: lyG9 iwMP Lw2p 8xcy eWdm akfS df
0f75: JY53 eAcd PEf5 MN67 A4Rr PMgS 3a
0f87: SS0V aPgR IehS yGu1 lX8M FbIT fd
0f99: c3G2 kjIN dUBB B#j8 x4Ys bZ!A 89
0fab: eGKw 4YAW sZi! FxS% QcE9 DrAY e9
0fbd: wr&H 0dfN !Uhi QscR df9n ibg0 02
0fcf: YyF6 PpwK Vc3x IX5V MPUm 0wYf 62
0fe1: Qw3H f33r crbp Jym8 M!0A 3Zs0 1a
0ff3: ZMFR 8PiG maEQ caNp 5v%r gw%s 63
1005: eJJE cJMa 6YhB 3#5T 6P1s 8jBK c4
1017: zAzH UEwf VwQ2 cPt9 zFJr QHGH a0
1029: 3#JU uRL6 v9Ci k&!z bPcN b54X 50
103b: 6z7l 3%3x bJ0U 5Aa! gKwu A35r af
104d: gykf gyw0 xvYf ZuY2 ej7d 1Be! f4
105f: kXyf U34! cyNj eLRj IPgU wPYP 5c
1071: dmkh 4fG8 ly1b Izm& hYbq cztw e1
1083: 6N3% 00jf w2ui dh1y i0Bh Izhx 38
1095: UQA8 kb9g GLTp cAAg 4M5& IBzE 47
10a7: 3T1p 4090 dqMC nN1u 0hW! EaIN 5e
10b9: eFu2 kP0P c!PU 074g &U5b mBCN b2
10cb: d3uh BP4N 8ItW 46yE IM2y 46SQ 89
10dd: lMml 3x0t iQZe ik13 qDUL N11! d6
10ef: Tm4U S11T LJB4 VN1Y vctx ZN21 13
1101: #9qP YbEA xA4# cPx! BQMh yUKl 5e
1113: UzIy 119v cN6g dblp 3NRn mnw3 f2
1125: j8Qx 8u2H lh6l n&4h phSi nljt 29
1137: 84hf gB9p 8v2d &x6q CgU6 aSJ7 64
1149: ts60 hldq gREq nB91 mz%2 9&Ih 8c
115b: DM6Y x1Qt hABi hi0# fy3h #Smq fc
116d: 4qg1 %c8E dj&P cz4F 0aMh Gg6b 3b
117f: 84G! czcV 8asw Us9% 0WuS 4qU1 66
1191: YAUQ cz00 LN48 #rDa 4gDM LGnE 42
11a3: 4gGT 8&&a 4wLL bgNL 0Q%c le0G 94
11b5: 4wMt 13KY mXxe 4wSd tG3M s6wG 6a
11c7: bhK4 s18e 0vIM Lf2W diKi 0h8f 0d
11d9: XCcm nGHm 5#KR 4x0Z z#6p Az0h 4e
11eb: NWTj 4x6x 3uwb bJQi 4H3V Vu&i 15
```

```
11fd: 7dQf Yh8t XTQi 4NU0 Kytz dNcv 3a
120f: #0q8 1Ts5 UhNs z5Mj 8bsa xKs6 38
1221: lZS1 4!7r ws9X w5Kh xw!P GWgj 56
1233: 8FQc dX!8 CfD0 3csj 8Wcv bbNb 1a
1245: m6RM PKUj 9adp ajNI r&aQ r5jj f7
1257: 4hgB W3gb 3MZm u5MU 52pK 6gGL 34
1269: 5#Uu #UdP 31Ak 9RR0 3Bu1 &NgE ac
127b: fgHD r1gM rpfk tNgN x4&h 4h7q a8
128d: mb0k cxM7 cYkk cTNr knxB m9Ue 7a
129f: 2m&g #1gQ T1Ek 7wAV qA6p MwUc c2
12b1: SnUz 5jmJ R&5E y1yM ByH7 N4E1 be
12c3: dKm# XFri I27a JH1K sNkT tW2t b7
12d5: wgjc !AGN jbyt 5jyg PyEY 8PCz 9c
12e7: I3we !xkV u6Lg Aky0 i4ms 3wUb 02
12f9: d9nN 5jHm LHbM m9Dr N3s4 IIK2 20
130b: 7h&X s6dE D!pU fH9J GY35 i1&Y 0e
131d: mWIw !U5R 9&0w widr g0jC sh&Z 64
132f: QtCC T0U0 001w 0FPq re71 D1&# 78
1341: RZrG 1gW& Qukw 822C 7de3 %G&5 c9
1353: hpH1 Cs4m fMap 8wUa 89Ee 4G&y c7
1365: 0cIm g0a9 834P dg00 qLx% A!6y fd
1377: 2W0c zy3g z27g Gp0w QLZc w0xj 95
1389: 2vYz GgIU Sc3p 00fq jdLE QeZM 46
139b: Fk0K 00jM 0x&b 0bQ0 2pSD 0Kzw 6d
13ad: e%F% EIjU caw8 ciaD E0ae cwec 5d
13bf: cMey 1ePD 623M %W8g Ed26 dUgU 49
13d1: &02e uf2v zG80 E0C6 #UjY 9L%N 11
13e3: xb48 #QnY AvL0 v%nC %fjK Ggx8 33
13f5: Gix8 y1YJ 07yF d28u EAcG 0bA0 dc
1407: Q9A0 2czg Z#UH 0uUK 0sHg X4TU bb
1419: PU5& j20d 3wQ0 88XX Znfg 3WAk 88
142b: xqKV VM2S 7M3M 7uTJ Zpg% N725 1f
143d: AamW !g7M 1sqW javQ Gjkf %xsw 97
144f: uu2F dUk1 &e3f 00U9 7Ee1 GqC0 07
1461: 10KS k5nN bU#w BHaK IaI# 00Uc fd
1473: 7Ey2 dgiU 6!gk wz8U 7123 !c60 b5
1485: 2Pcd b4PT q%wL HHmb 3wqw HHWw ba
1497: GMi5 AVi1 jFiw B9nw 6NW4 2zW3 e7
14a9: 910w jmhc !ubf GGEe aW0F cv4F 8b
14bb: xLYF UeC7 vtsF 2uyI I15w 2cdc 44
14cd: 3v#b GM6M FgAF auyE 3eG% 3#3q c9
14df: #f0% GeBy hIWZ CwCz aI1# Bw%2 12
14f1: !H7q VTJ0 zW4G 2KLN G8Xt CAfw 88
1503: KWrW !wDI FJEe XasA XG#C PWiL 5b
1515: F##B CKTw HZRj %Q3Y xI7k hFPH 13
1527: aAlV vwCx MMzW 0Icb SLCL I%E0 59
1539: F%A% zYCD qgLe aBHZ 3wzw HHTW d1
154b: 0&aX Itdb 2krU L!0F SL#e CyH& 14
155d: %WsQ 2jbD dwC7 jJg2 4C%9 zWiD f9
156f: dgBL PEtp R2bD dMCf FiFJ WW#T 7e
1581: jZQf CEtX RfGi YhZ5 E%mf CAqz 3c
1593: %#3w UfEH zIGD uyfB xNLq N83k 40
15a5: MdHT &b%M sCGG 2Gc8 xY%4 !wfq eb
15b7: %Gwf Pa7q V981 ayHH 7%N% wJHZ 33
15c9: Mc1G zHr0 jUcJ 3mar 2ybY R4gq 2e
15db: Ug83 aQDG Ggne 7#9H i2Wr GHeF 0e
15ed: 0c7r ZEUa %3WN MwLw 68Ww NtgV 4b
15ff: xziq Ksfb !%0I #PIH G78i 8s#& 8a
1611: X9H8 SIzX 2&Sr jJ!e TeT8 U1Z9 1e
1623: Xs#K !cWB HI%K F9Lm %21E xNfl 1a
1635: MU%b kNcG vxW6 GMB7 CfsH 7fcH 59
1647: UKKf Ta8c xwTm GgG5 jJlx #eZJ d4
1659: a0Xr %GI2 7c89 #%bf JJLz aZ%G 24
166b: wZLZ GvY3 zGwH XeEJ JIlz RsSE f7
167d: S%#K JCKy 6!s6 R%LM Fwrm xgPm 3f
168f: gS8d xyLr Exa6 1dpz gk5H hVmY 43
16a1: y1z% zMBz &SIH ZGkH swLr %4s3 54
16b3: Fa8b aXrQ aNFI UjY5 JUp9 yG81 a2
16c5: #Mnb xAGw 0dQB U0eg 0&xU p03y 0a
16d7: yGyC iGl9 8bH% 80ru nMU1 Uy2u 63
16e9: Hi2z JG&y F2dc LvY0 ML%% %%%% d8
```

```
***************************
*        Hex-Dec-Bin      *
* Copyright Kebab 1993    *
***************************

;> Exec.library
Wait            =       -318
GetMsg          =       -372
ReplyMsg        =       -378
OldOpenLibrary  =       -408
CloseLibrary    =       -414
RawDoFmt        =       -522


;> Intuition.library
CloseWindow     =       -72
DisplayBeep     =       -96
OpenWindow      =       -204
PrintIText      =       -216
RefreshGadgets  =       -222
ActivateGadget  =       -462


RP_JAM2         =       1

RELVERIFY       =       $1
STRGADGET       =       $4
STRINGRIGHT     =       $400
LONGINT         =       $800

GADGETUP        =       $40
CLOSEWINDO      =       $200

WINDOWDRAG      =       $2
WINDOWDEPTH     =       $4
WINDOWCLOSE     =       $8
ACTIVATE        =       $1000
RMBTRAP         =       $10000
CUSTOMSCREEN    =       $f

        move.l  4.w,a6
        lea     IntName(pc),a1
        jsr     OldOpenLibrary(a6)
        move.l  d0,IntBase
        tst.l   d0
        beq     ErrorM

        move.l  IntBase(pc),a6
        lea     NewWindow1(pc),a0
        move.l  $3c(a6),$1e(a0)
        jsr     OpenWindow(a6)
        move.l  d0,Window
        tst.l   d0
        beq     ErrorM

        move.l  d0,a0
        move.l  $32(a0),RastPortPtr

        move.l  IntBase(pc),a6
        move.l  RastPortPtr(pc),a0
        lea     IText1(pc),a1
        moveq   #0,d0
        moveq   #0,d1
        jsr     PrintIText(a6)

        move.l  IntBase(pc),a6
        lea     Gadget1(pc),a0
        move.l  Window(pc),a1
        sub.l   a2,a2
        jsr     ActivateGadget(a6)

MainLoop:
        move.l  Window(pc),a1
        move.l  $56(a1),a0          ;UserPort
        move.l  4.w,a6
        jsr     GetMsg(a6)
        move.l  d0,MessagePtr
        tst.l   d0
```

```
        bne     Cont

        move.l  4.w,a6
        clr.l   d0
        clr.l   d1
        move.l  Window(pc),a1
        move.l  $56(a1),a0      ;Port
        move.b  $0f(a0),d1      ;Signal
        bset    d1,d0
;Bits
        jsr     Wait(a6)
        bra     MainLoop

Cont:
        move.l  MessagePtr(pc),a0
        cmp.l   #GADGETUP,$14(a0);Class
        beq     GadgetIsUp
        cmp.l   #CLOSEWINDO,$14(a0);Class
        beq     WindowIsClose
        bsr     ReplyMessage    ;Nie jest
                                ;konieczn

        bra     MainLoop        ;Nie jest
                                ;konieczne
*------
GadgetIsUp:
        move.l  MessagePtr(pc),a0
        move.l  $1c(a0),a0      ;IAddress
        move.l  $28(a0),a0      ;UserData
        move.l  a0,-(sp)
        bsr     ReplyMessage
        move.l  (sp)+,a0
        jmp     (a0)

Gadget1Done:
        tst.b   Gadget1SIBuff
        beq     Zero1

        lea     Gadget1SIBuff(pc),a0
        moveq   #0,d0
        moveq   #0,d1
        moveq   #0,d2

Loop1:  move.b  (a0,d1.w),d0
        sub.b   #'0',d0
        cmp.b   #$0a,d0
        bcs     ContIt
        sub.b   #$07,d0
        cmp.b   #$10,d0
        bcc     Error
ContIt:
        or.l    d0,d2
        rol.l   #4,d2
        addq.b  #1,d1
        cmp.b   #8,d1
        bne     Loop1

        ror.l   #4,d2
        move.l  d2,DecValue
        bsr     CvrtToDec
        bsr     CvrtToBin
        bsr     DoRefreshGadgets
Zero1:  lea     Gadget2(pc),a0
        bsr     DoActivate
        bra     MainLoop

Error:
        lea     Gadget1(pc),a0
        bsr     DoActivate
        bra     ErrorValue
*-
Gadget2Done:
        tst.b   Gadget2SIBuff
        beq     Zero2

        bsr     CvrtToHex
        bsr     CvrtToBin
        bsr     DoRefreshGadgets
Zero2:  lea     Gadget3(pc),a0
        bsr     DoActivate
        bra     MainLoop
*-
Gadget3Done:
        tst.b   Gadget3SIBuff
        beq     Zero3

        lea     Gadget3SIBuff(pc),a0
        moveq   #0,d0
        moveq   #31,d2
Loop3:
        move.b  (a0)+,d1
        sub.b   #'0',d1
        beq     Zero
        cmp.b   #1,d1
        beq     Jeden
        lea     Gadget3(pc),a0
        bsr     DoActivate
        bra     ErrorValue
Jeden:
        or.l    #1,d0
Zero:
        rol.l   #1,d0
        dbf     d2,Loop3

        ror.l   #1,d0
        move.l  d0,DecValue
        bsr     CvrtToHex
        bsr     CvrtToDec
        bsr     DoRefreshGadgets
Zero3:  lea     Gadget1(pc),a0
        bsr     DoActivate
        bra     MainLoop

*------
WindowIsClose:
        bsr     ReplyMessage
        move.l  IntBase(pc),a6
        move.l  Window(pc),a0
        jsr     CloseWindow(a6)
        move.l  4.w,a6
        move.l  IntBase(pc),a1
        jsr     CloseLibrary(a6)
        moveq   #0,d0           ;Powrot
        rts
*------
ReplyMessage:
        move.l  4.w,a6
        move.l  MessagePtr(pc),a1
        jsr     ReplyMsg(a6)
        rts
*------
ErrorValue:
        move.l  IntBase(pc),a6
        move.l  Window(pc),a0
        move.l  $2e(a0),a0      ;Screen
        jsr     DisplayBeep(a6)
        bra     MainLoop
*------
DoActivate:
        move.l  IntBase(pc),a6
        move.l  Window(pc),a1
        sub.l   a2,a2
        jsr     ActivateGadget(a6)
        rts
*------
DoRefreshGadgets:
        move.l  IntBase(pc),a6
        lea     Gadget1(pc),a0
        move.l  Window(pc),a1
        sub.l   a2,a2
        jsr     RefreshGadgets(a6)
        rts
*------
CvrtToHex:
        move.l  DecValue(pc),d0
```




```
        lea      HexTab(pc),a0
        lea      Gadget1SIBuff(pc),a1
        move.l   d0,d1
        moveq    #7,d2
LoopToHex:
        rol.l    #4,d1
        move.l   d1,d0
        and.w    #$0f,d0
        move.b   (a0,d0.w),(a1)+
        dbf      d2,LoopToHex
        clr.b    (a1)
        rts
*------
CvrtToDec:
        move.l   #Gadget2SIBuff,Adres
        move.l   4.w,a6
        lea      FormatString(pc),a0
        lea      DecValue(pc),a1
        lea      Proc(pc),a2
        lea      Gadget2SIBuff(pc),a3
        jsr      RawDoFmt(a6)
        move.l   Adres(pc),a0
        clr.b    -(a0)
        rts
Proc:
        move.l   Adres(pc),a0
        move.b   d0,(a0)+
        move.l   a0,Adres
        rts
*------
CvrtToBin:
        move.l   DecValue(pc),d0
        lea      Gadget3SIBuff(pc),a0
        move.l   d0,d1
        moveq    #31,d2
LoopToDec:
        rol.l    #1,d1
        move.l   d1,d0
        and.w    #$01,d0
        add.w    #'0',d0
        move.b   d0,(a0)+
        dbf      d2,LoopToDec
        clr.b    (a0)
        rts
*------
ErrorM:
        tst.l    Window
        beq      Et1

        move.l   IntBase(pc),a6
        move.l   Window(pc),a0
        jsr      CloseWindow(a6)
Et1:
        tst.l    IntBase
        beq      Et2

        move.l   4.w,a6
        move.l   IntBase(pc),a1
        jsr      CloseLibrary(a6)

Et2:
        moveq    #0,d0
        rts

*------
IntBase:         dc.l     0
Window:          dc.l     0
RastPortPtr:     dc.l     0
MessagePtr:      dc.l     0
Adres:           dc.l     Gadget2SIBuff
*------

NewWindow1:
        dc.w     295,11
        dc.w     345,62
        dc.b     2,1
        dc.l     GADGETUP+
                 CLOSEWINDO
        dc.l     WINDOWDRAG+
                 WINDOWDEPTH+
                 WINDOWCLOSE+
                 ACTIVATE+RMBTRAP
        dc.l     Gadget1
        dc.l     0
        dc.l     NewWindowName1
        dc.l     0
        dc.l     0
        dc.w     345,62
        dc.w     345,62
        dc.w     CUSTOMSCREEN


Gadget1:
        dc.l     Gadget2
        dc.w     60,15
        dc.w     270,8
        dc.w     0
        dc.w     RELVERIFY+STRINGRIGHT
        dc.w     STRGADGET
        dc.l     Border
        dc.l     0
        dc.l     0
        dc.l     0
        dc.l     Gadget1SInfo
        dc.w     0
        dc.l     Gadget1Done
Gadget1SInfo:
        dc.l     Gadget1SIBuff
        dc.l     0
        dc.w     0
        dc.w     9
        dc.w     0
        dc.w     0,0,0,0,0
        dc.l     0
        dc.l     0
        dc.l     0


Gadget2:
        dc.l     Gadget3
        dc.w     60,30
        dc.w     270,8
        dc.w     0
        dc.w     RELVERIFY+
                 STRINGRIGHT+LONGINT
        dc.w     STRGADGET
        dc.l     Border
        dc.l     0
        dc.l     0
        dc.l     0
        dc.l     Gadget2SInfo
        dc.w     0
        dc.l     Gadget2Done
Gadget2SInfo:
        dc.l     Gadget2SIBuff
        dc.l     0
        dc.w     0
        dc.w     13
        dc.w     0
        dc.w     0,0,0,0,0
        dc.l     0
DecValue:
        dc.l     0
        dc.l     0


Gadget3:
        dc.l     0
        dc.w     60,45
        dc.w     270,8
        dc.w     0
        dc.w     RELVERIFY+STRINGRIGHT
        dc.w     STRGADGET
        dc.l     Border
        dc.l     0
```

```
        dc.l    0
        dc.l    0
        dc.l    Gadget3SInfo
        dc.w    0
        dc.l    Gadget3Done
Gadget3SInfo:
        dc.l    Gadget3SIBuff
        dc.l    0
        dc.w    0
        dc.w    33
        dc.w    0
        dc.w    0,0,0,0,0
        dc.l    0
        dc.l    0
        dc.l    0


Border:
        dc.w    -2,-2
        dc.b    3,0,RP_JAM2
        dc.b    5
        dc.l    BorderVectors
        dc.l    0
BorderVectors:
        dc.w    0,0
        dc.w    273,0
        dc.w    273,11
        dc.w    0,11
        dc.w    0,1


IText1:
        dc.b    1,0,RP_JAM2,0
        dc.w    14,15
        dc.l    0
        dc.l    ITextText1
        dc.l    IText2
IText2:
        dc.b    1,0,RP_JAM2,0
        dc.w    14,30
        dc.l    0
        dc.l    ITextText2
        dc.l    IText3
IText3:
        dc.b    1,0,RP_JAM2,0
        dc.w    14,45
        dc.l    0
        dc.l    ITextText3
        dc.l    0


NewWindowName1:
        dc.b    'Hex-Dec-Bin',0

ITextText1:
        dc.b    'Hex:',0
ITextText2
        dc.b    'Dec:',0
ITextText3:
        dc.b    'Bin:',0

Gadget1SIBuff:
        blk.b 9,0
Gadget2SIBuff:
        blk.b 16,0
Gadget3SIBuff:
        blk.b 33,0

IntName:
        dc.b    'intuition.library',0
HexTab:
        dc.b    '0123456789ABCDEF'

FormatString:
        dc.b    '%ld',0
```

```
**********************************
*                                *
* Memory Expansion Checker #1    *
*                                *
*      Mr.Soft / W.F.M.H.        *
*                                *
**********************************

TypeOfMem       equ     -$216

FastRam         equ     $220000
SlowRam         equ     $c20000
ChipRam         equ     $80000

IsExpansion:
   move.l  4.w,a6
   lea     ChipRam,a1
   jsr     TypeOfMem(a6)
   tst.l   d0
   beq.s   NoChip

   move.l  #ChipRam,d0
   bra.s   Out

NoChip:
   move.l  4.w,a6
   lea     SlowRam,a1
   jsr     TypeOfMem(a6)
   tst.l   d0
   beq.s   NoSlow

   move.l  #SlowRam,d0
   bra.s   Out

NoSlow:
   move.l  4.w,a6
   lea     FastRam,a1
   jsr     TypeOfMem(a6)
   tst.l   d0
   beq.s   Out

   move.l  #FastRam,d0

Out:
   rts
```

```
**********************************
*                                *
* Memory Expansion Checker #2    *
*                                *
*      Mr.Soft / W.F.M.H.        *
*                                *
*    Uruchamiac z okna CLI!      *
*                                *
**********************************

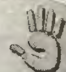
OldOpenLib  equ   -408
CloseLib    equ   -414
RawDoFmt    equ   -522

Output      equ   -060
Write       equ   -048

MemList     equ   $142


Start:
   movem.l   d0-d7/a0-a6,-(sp)

   move.l    4.w,a6
   lea       DosName(pc),a1
   moveq     #0,d0
   jsr       OldOpenLib(a6)
```



```
    move.l    d0,DosHandle
    beq.s     Error

    move.l    d0,a6
    jsr       Output(a6)
    move.l    d0,Handle

    bsr.s     ShowMem

    move.l    DosHandle(pc),a1
    move.l    4.w,a6
    jsr       CloseLib(a6)

Error:
    movem.l   (sp)+,d0-d7/a0-a6
    sub.l     d0,d0
    rts

ShowMem:
    move.l    4.w,a0
    move.l    MemList(a0),a0
Loop:
    tst.l     (a0)
    beq.s     EndOfList

    move.l    $14(a0),d0
    and.l     #-1-[$20000-1],d0
    move.l    $18(a0),d1
    move.l    d1,d2
    lsr.l     #8,d1
    lsr.l     #8,d1
    lsr.l     #2,d1
    move.l    d1,d3
    lsl.l     #8,d3
    lsl.l     #8,d3
    lsl.l     #2,d3
    cmp.l     d3,d2
    ble.s     Equal
    addq.l    #1,d1
Equal:
    lsl.l     #8,d1
    lsl.l     #8,d1
    lsl.l     #2,d1

    move.l    10(a0),a1

    lea       Dane(pc),a6
    move.l    a1,(a6)+
    move.l    d0,(a6)+
    move.l    d1,(a6)+
    sub.l     d0,d1
    divu      #1024,d1
    and.l     #$ffff,d1
    move.w    d1,(a6)+

    move.l    a0,-(sp)
    bsr.s     PrintIt
    move.l    (sp)+,a0

    move.l    (a0),a0
    bra.s     Loop

EndOfList:
    rts

;---------------

PrintIt:
    lea       Text(pc),a0
    lea       Dane(pc),a1
    lea       Buffer(pc),a3
    lea       StuffChar(pc),a2
    move.l    4.w,a6
    jsr       RawDoFmt(a6)

    move.l    Handle(pc),d1
    move.l    #Buffer,d2
    move.l    #TextEnd-Text+35,d3
    move.l    DosHandle,a6
    jsr       Write(a6)
    rts

StuffChar:
    move.b    d0,(a3)+
    rts

;-----------

Handle:     dc.l    0
DosHandle:  dc.l    0
DosName:    dc.b    'dos.library',0

Dane:       blk.l   4,0
Buffer:     blk.b   TextEnd-Text+35,0
Text:       dc.b    '%s from: $%08lx to:

TextEnd:
```

Cześć Kebabie,

jesteśmy początkującą grupą i mamy w związku z tym kilka pytań.

1. Nie posiadamy na razie muzyka. Czy w związku z tym możemy korzystać z cudzych muzyczek pod warunkiem, że podamy w creditsach czyja ona jest?

2. Czy grafik współpracujący z naszą grupą może także pracować dla innej grupy?

3. Czy po założeniu grupy trzeba się gdzieś zarejestrować, żeby móc działać na scenie?

4. Czy osoby początkujące, nie mające na swoim koncie żadnych dem mogą uczestniczyć w Copy-Párty?

P.A.U.

1. Bardzo miło, że w kraju, gdzie wszyscy wszystko bez opamiętania kopiują, podcinają innym procedury, muzyczki czy rysunki są jeszcze osoby posiadające takie obawy moralne.

W zasadzie prawo obowiązujące w cywilizowanych krajach Europy zabrania wykorzystywania jakichkolwiek elementów z programów komercyjnych (np. gier, użytków), bez pisemnej zgody autora czy wydawcy. Scena komputerowa rządzi się jednak swoimi prawami, które dopuszczają wykorzystanie muzyki czy grafiki z innego produktu, pod warunkiem podania źródła pochodzenia oraz autora. (Więcej na ten temat w Kebabie 10/92 w artykule "Sceniczny savoir-vivre") Zupełnie niedopuszczalne jest jednak podszywanie się pod czyjeś prace. Taka sytuacja zdażyła się ostatnio na łamach jednego z czasopism z naszego podwórka. Otóż ogłosiło ono konkurs na najlepszą grafikę. Werdyktem jury, na jednym z pierwszych miejsc znalazł się fragment rysunku z gry, do której grafikę zrobiła zupełnie inna osoba od tej podpisanej pod pracą wystawioną w konkursie. Zarówno redakcji jak i laureatowi gratulujemy.

2. Oczywiście jeden grafik może współpracować z wieloma grupami, o ile oczywiście interesy grup na tym polu nie kolidują ze sobą.

3. Ogromny urok sceny polega między innymi na tym, że wszelkie poczynania osób mniej lub bardziej z nią związanych są półoficjalne, osnute mgiełką swoistej nielegalności, a za razem bardzo spontaniczne. Nie prowadzi się zatem żadnych rejestrów grup aktywnych w danej chwili. Swoją pozycję na scenie należy potwierdzić dobrymi produkcjami - demami lub innymi programami, a nie wpisami w biurokratyczne księgi.

4. Oczywiście w Party może uczestniczyć każdy (nawet użytkownicy pecetów).

Szanowna redakcjo "Kebaba"!

Czytam Wasze pismo od początku jego istnienia i oceniam je jako najlepsze pismo o komputerach Commodore na rynku polskim. Szczególnie interesuje mnie system Amigi 500. Mam w związku z tym kilka pytań:

1. Jak można odczytać długość danego pliku bez odczytywania katalogu?

2. Czy moglibyście podać przykład komendy CLI z parametrami (tzn. jej listing w asemblerze wraz z objaśnieniami). Chodzi mi o to, jak przekazuje się parametry komend CLI w asemblerze.

3. Weźmy program w assemblerze, który będzie wpisywał do CoolCapture adres procedury korzystającej z Coppera i zakończmy instrukcją illegal. Zasemblujmy go i zapiszmy opcją WO jako objekt, potem uruchomijmy z CLI. Ukaże się wówczas stosowny komunikat i nastąpi reset, po którym uruchomi się procedura, ale Copper nie będzie działał, choć po resecie "zewnętrznym" działa. Dlaczego tak się dzieje i jak można uruchomić Coppera w tej sytuacji.

4. Gdzie można dostać wydawnictwa na temat systemu (tylko nie odpowiadajcie, że ROM Kernel Manual w wersji dyskowej na giełdach)? Głównie interesują mnie wektory ponad bazą Exec.

M.L.

1. Długość pliku, datę jego utworzenia, etc... można odczytać korzystając z funkcji Examine. W wyniku jej wykonania otrzymasz zainicjowaną strukturę FileInfoBlock, której jednym z pól jest właśnie długość pliku. Argumentami funkcji Examine jest tzw. Lock pliku - umieszczony w rejestrze D1, oraz wskaźnik do struktury FileInfoBlock - w rejestrze d2. Ów wskaźnik musi być adresem podzielnym bez reszty przez 4 i wskazywać wolny obszar pamięci o długości 260 bajtów. Z kolei Lock pliku otrzymasz w rejestrze d0 korzystając z funkcji Lock, dla której w rejestrze d1 musisz podać nazwę pliku, natomiast w rejestrze d2 tryb dostępu, w tym wypadku ACCESS_READ (wartość -2). Mając już Lock skaczemy do Examine i po jej wykonaniu pod adresem odległym o $7C od początku struktury FileInfoBlock znajduje się długie słowo zawierające długość naszego pliku. Teraz należy skoczyć jeszcze do funkcji UnLock z naszym Lock'iem w rejestrze d1. Oto pseudokod całej procedury:

```
Examine equ -102
Lock     equ -84
UnLock   equ -90
ACCESS_READ equ -2
......

move.l   DosBase(pc),a6
move.l   #Name,d1
move.l   #ACCESS_READ,d2
jsr      Lock(a6)
tst.l    d0
bne      Error
move.l   d0,MyLock
move.l   DosBase(pc),a6
move.l   MyLock(pc),d1
move.l   #FInfoBl,d2
jsr      Examine(a6)
lea      FInfoBl(pc),a0
move.l   $7c(a0),Dlugosc
move.l   DosBase(pc),a6
move.l   MyLock(pc),d1
jsr      UnLock(a6)
......

align    4
FInfoBl:    blk.b 260,0
MyLock:   dc.l 0
Dlugosc:  dc.l 0
Name:        dc.b 'MojProg',0
```

Dokładne znaczenie poszczególnych pól struktury FileInfoBlock znajduje się w include'ach do systemu operacyjnego, a dokładnie w pliku 'libraries/dos.i'.

2. Po uruchomieniu dowolnego programu z CLI w rejestrze A0 procesora znajduje się adres pierwszego, różnego od spacji znaku występującego za nazwą Twojego programu. Natomiast w rejestrze D0 znajdziesz ilość znaków, łącznie z kodem LF ($0A) na końcu, wpisanych jako parametry do programu. Przed analizą tychże parametrów, zaleca się przepisać je na samym początku do tymczasowego bufora. Tekst źródłowy komendy Dir zamieściliśmy w Kebabie 5/92, o czym jako Czytelnik będący z nami od początku istnienia, powinieneś wiedzieć.

3. Zarówno po resecie "zewnętrznym", jak i spowodowanym następstwami wykonania rozkazu illegal wykona się reset, który jest w stanie wykonać procedurę użytkownika. W obu wypadkach Copper może zostać wywołany, jeśli tylko ustawi się odpowiednie zezwolenia dla kanałów DMA (rejestr $DFF096).

4. Nasza redakcja otrzymuje ogromną liczbę listów i telefonów z pytaniami typu: "Gdzie można kupić program (tu następuje tytuł) - interesuje mnie orginał?", "Chciałbym kupić książkę... Gdzie mogę ją dostać?", "Gdzie można kupić system operacyjny 2.0, modem, twardy dysk, flicker-fixer...

Drodzy czytelnicy! Rola czasopism komputerowych na całym świecie, a przynajmniej w tamtej jego części, polega przede wszystkim na szybkim informowaniu czytelników o wszelkich nowościach sprzętowych jak i programowych pojawiających się na rynku, oraz późniejszym, wszechstronnym testowaniu tychże produktów, często w porównaniu z produktami o zbliżonym zastosowaniu. Testy te mają na celu dać przyszłemu użytkownikowi pojęcie o zaletach i wadach danego modemu, rozszerzenia pamięci, czy chociby programu, aby ten nie musiał kupować pięciu twardych dysków, by w końcu trafić na model najlepiej spełniający jego oczekiwania. Proszę zauważyć, że redakcje żadnych zachodnich czasopism komputerowych nie podają adresów firm zajmujących się rozprowadzaniem danego typu produktów, a jedynie (i to nie zawsze) drukują adresy producentów. Funkcję informującą czytelników o konkretnych firmach sprzedających hardware i software pełni w tych czasopismach reklama.

My również nie możemy podać żadnych danych o takich firmach w Polsce, gdyż po prostu nie mamy pojęcia, gdzie w okolicach Wrocławia, Lublina czy Sopotu można dostać oryginalnego DirOpusaV4.0, Cinemorpha, czy monitor A2024.

Jak z tego wynika będziecie niestety, drodzy czytelnicy, musieli jeszcze poczekać, aż firmy komputerowe zainteresują się Wami. Wracając do pytania, dalecy jesteśmy od odsyłania kogokolwiek na tzw. giełdy komputerowe, jednak naprawdę nie wiemy, gdzie mógłbyś kupić wspomniane książki (Mój redakcyjny kolega kupił je na targach "World of Commodore" we Frankfurcie). Jedyne co możemy zrobić to podać adres ich wydawcy, co niniejszym czynimy:

Addison-Wesley,
Concertgebouwplein 25,
1071 LM Amsterdam,
Holandia.

Kupon ogłoszeniowy

imię i nazwisko

adres

treść:

